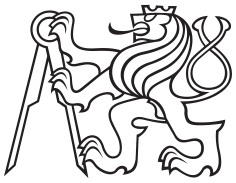
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Robustní postupy řízení formací robotů

Martin Novotný

Vedoucí: Ing. Libor Přeučil, CSc.
Obor: Robotika
Studijní program: Kybernetika a robotika
Květen 2017

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Liboru Přeučilovi, CSc. za podnětné připomínky a užitečné rady. Dále bych rád poděkoval Ing. Janu Chudobovi, Dr. Gaël Pierre Marie Écorchardovi a Ing. Martinu Dörflerovi za věcné rady při práci na tomto projektu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne .....

.....
podpis autora práce

Abstrakt

Předmětem této práce je vytvořit algoritmus pro řízení formace mobilních pozemních robotů na základě známé relativní polohy. Úkolem formace je následovat vedoucí robot, který bude ovládán člověkem. Řídící algoritmus by také měl být vybaven funkcionalitou vyhýbání se překážkám. Pro stabilizaci formace během jízdy byly zvoleny algoritmy využívající virtuální pružiny a potenciálová pole. Oba algoritmy byly implementovány v simulačním prostředí V-rep a následně byly obě metody otestovány sérií experimentů. Ty potvrdily schopnost řídit formaci oběma metodami.

Klíčová slova: stabilizace formace, robot, následování, virtuální pružiny, potenciálová pole, vyhýbání se překážkám

Vedoucí: Ing. Libor Přeučil, CSc.

Abstract

The subject of this work is to create control algorithm for mobile ground robot formation based on relative location. The formation task is to follow a leader robot, which is controled by a human operator. Control algorithm incorporates also collision avoidance. Robot swarm stabilization is accomplised via virtual springs algorithm and potential field approach. Both of these were implemented in a simulation environment using V-rep tool and then tested in series of experimetns. The tests confirmed capability of the approach to control formation movement for both methods.

Keywords: formation stabilization, swarm robotics, following, virtual spring, potential field, collision avoidance

Title translation: Robust Control of Robot Formations

Obsah

Obrázky

Tabulky

Úvod

V současné době, kdy industrializace dosahuje nové kvality ve formě Průmyslu 4.0 a ve výrobních procesech jsou implementovány složitější a sofistikovanější technologie, je třeba výrobní proces řešit komplexně s cílem maximální koordinace při všech úkonech s minimální časovou náročností. Osamocené robotické systémy již nedokáží naplnit požadavky na rychlost, přesnost, kvalitu a požadavky trhu. V současnosti je třeba do výroby implementovat sofistikovanější automatické robotické procesy. Východiskem v moderních a nadčasových společnostech je kooperace sladěných robotických systémů, které pracují v rámci jedné ucelené a propojené skupiny, ve které se navzájem sdílejí informace – „sharing and pooling“. Pro řízení těchto skupin se můžeme inspirovat ve světě zvířat či hmyzu, kde se takové skupiny vyskytují velmi často, jedná se například o hejna ptáků nebo ryb. Robotická hejna (swarm), nabízí řešení pro složité výrobní procesy, kde je kladen důraz na rychlost a vysokou kvalitu. Oproti ruční výrobě s využitím jednotlivých robotů, poskytují robotická hejna komplexní sofistikované řešení.

Tato bakalářská práce se konkrétně zabývá vytvořením řídícího algoritmu pro swarm mobilních pozemních robotů. Cílem projektu je vytvořit autonomní robotické hejno, jenž bude schopno sledovat vedoucí robot (leader) řízený člověkem. Řešení bude využívat kamery pro zjištění relativních poloh a zároveň bude obsahovat funkcionalitu vyhýbání se menším překážkám (collision avoidance) se schopností následně se zařadit zpět do formace.

Představme si například situace sekání trávy na golfových hřištích, úklid sněhu na letištních plochách, sklizeň obilí na polích nebo čištění velkých prostor. V každém z těchto případů je využívána řada strojů, které se pohybují po podobných trajektoriích. Každý z těchto strojů je nutné řídit. To dnes zajišťují kvalifikovaní pracovníci nebo se využívá řízení užitím GNSS (GPS)

signálu. Použití lidského faktoru s sebou přináší určitá rizika oproti použití automatického řízení. Využití řízení užitím GPS signálu nemusí být ale vždy vhodné, například kvůli špatnému počasí, v zalesněných oblastech či v budovách. Použití kamery pro určení polohy by naopak v případech, kdy selhává lokace pomocí GPS, bylo kvalitní a komplementární náhradou.

V první kapitole této práce jsou shrnuty obecně používané metody pro udržování formací robotů. Jsou analyzovány možné tvary formací a jejich vnitřní struktura v souvislosti s jejich využitím. Ve druhé kapitole je rozebrán princip vytváření collision avoidance pro dvě vybrané metody. Následně jsou také vloženy ukázky pseudokódu důležitých částí navrhovaných algoritmů. Ve třetí kapitole jsou ukázány možnosti testování kinematické stability formace v simulačním prostředí pro různé situace a z různých pohledů. V poslední kapitole je stručně popsán experiment s reálnými roboty se všemi jeho důležitými součástmi.

Kapitola 1

Teorie, metody a principy

1.1 Požadavky na algoritmus

Řídicí algoritmus by měl být schopen stabilizovat formaci během jízdy a vyhnout se menším překážkám. Prostřednictvím kamery a speciálních značek, které zajistí vzájemné měření relativní polohy jednotlivých robotů, by měly následníci sledovat vedoucí robot, který bude ovládán člověkem.

Tvar formace musí být přizpůsoben dostupným technickým prostředkům a použitému algoritmu řízení. K dispozici je laserový dálkoměr a černobílá kamera. S těmito prostředky jsme schopni určit vzájemnou relativní polohu vedoucího robotu, jednotlivých členů formace a překážek. Zjištěné relativní polohy byly následně použity v řídicím algoritmu pro výpočet požadované rychlosti jednotlivého robotu ve formaci.

Jednotlivé roboty jsou opatřeny laserovým dálkoměrem, který je použit pro vytvoření systému předcházení kolizím. Každý člen hejna by se tak měl být schopen vyhnout překážkám nezávisle na ostatních. Překážky by měly být menších rozměrů, jako jsou například stromy, sloupy nebo zúžená místa, apod., aby se vedoucí robot neztratil ze záběru kamery na příliš dlouhou dobu.

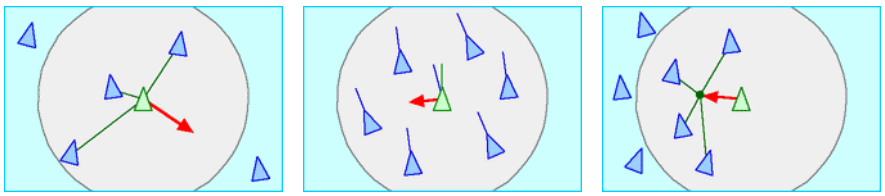
1.2 Principy

1.2.1 BOIDS

Craig Reynolds vytvořil v roce 1987 počítačovou simulaci letu ptáků v hejně [9], [20]. Pro svůj model se nechal inspirovat chováním ptáků a ryb; pracoval na základě vektorových výpočtů ve 3D prostředí. Název BOIDS vychází ze zkrácené verze pro Bird-Oid Objects. Tento model chování ve skupině vychází ze tří pravidel:

1. Vyhýbání se vzájemným kolizím - změna směru za účelem zamezení hromadění elementů skupiny na jednom místě.
2. Srovnání rychlostí - změna do stejného směru jako směr rychlosti sousedů.
3. Shlukování do středu - změna směru do středu hejna.

Tato pravidla pro řízení hejna se nazývají (1) separace (Separation), (2) zarovnání (Alignment) a (3) soudržnost (Cohesion).



(a) : Separace. **(b) :** Zarovnání. **(c) :** Soudržnost.

Obrázek 1.1: Pravidla shlukování BOIDS. Převzato z [19] 15.5.2017.

Separace představuje odpudivou sílu členů hejna. Zabraňuje hromadění členů na jednom místě a také brání srážkám. Zajišťuje rovnoměrné rozprostření členů v pracovním prostředí bez ohledu na rychlost. Závisí pouze na jejich poloze. Odpudivá síla k separaci jednotlivců se určí z váženého průměru polohy sousedů a směřuje od tohoto bodu (obrázek 1.1 a). Hodnota určující váhu je nepřímo úměrná vzdálenosti od souseda.

Zarovnání představuje sílu, která nutí člen hejna pohybovat se stejným směrem a stejnou rychlostí, jako jeho sousedé. Je závislá pouze na rychlosti sousedů a nebere ohled na jejich polohy. Síla také působí jako prevence proti

srážkám. Pokud člen udržuje stejný vektor rychlosti jako jeho sousedé (obrázek 1.1 b), pak je nepravděpodobné, že by došlo ke srážce. Udržování stejné rychlosti v ideálním případě zajistí prakticky neměnnou vzdálenost mezi členy hejna.

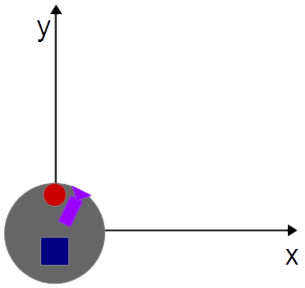
Soudržnost představuje sílu, která drží celé hejno pohromadě. Každý člen se snaží držet v okolí středu shluku (obrázek 1.1 c), tedy ve středu poloh okolních členů hejna. Nezabrání však rozdělení hejna před překážkou, protože tato síla daný člen směruje do středu jakéhosi lokálního hejna tvořeného jeho sousedy.

Tento algoritmus není zcela vhodný pro účel popsaný v úvodu práce, protože slouží spíše pro udržování robotů v hejnu, při kterém není potřeba přesně zajistit dané pozice robotů nebo tvar formace. V následujících dvou kapitolách jsou popsány dva vhodnější postupy splňující tuto podmínku.

1.2.2 Pružinový algoritmus

Algoritmus pružin je založen na stejném principu, na jaký využívá i klasická mechanická pružina [12]. Při kratší vzdálenosti, než je volná délka, se pružina roztahuje a naopak při větší vzdálenosti se stahuje.

Pro popis polohy robotu je použita pravotočivá kartézská souřadná soustava s počátkem umístěným ve středu příslušného robotu, přičemž daný robot se pohybuje v kladném směru osy y (obrázek 1.2).

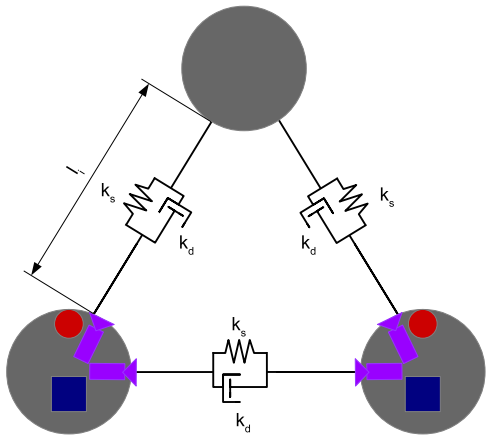


Obrázek 1.2: Souřadná soustava členu formace.

Je nutné předpokládat, že se vedoucí robot nebude pohybovat rychleji než je maximální dosažitelná rychlost ostatních robotů, jinak by problém neměl řešení. Dále je potřeba předpokládat nulové relativní zrychlení vzhledem k měřenému objektu. S využitím rovnice 1.1 [24] lze dopočítat rychlost, kterou je potřeba přiřadit robotu. Ta by měla dosahovat stejné hodnoty, jakou se pohybuje

leader.

Pomocí kamery lze zjistit relativní polohu vedoucího a konkrétního robotu, na němž se kamera nachází. Z této polohy je následně možné zjistit směrový vektor působení pružiny.



Obrázek 1.3: Struktura robotické formace s použitím pružinového algoritmu.

Obecný tvar rovnice popisující pružinovou závislost mezi konkrétním robotem a sledovaným objektem má tvar

$$m\ddot{\mathbf{x}} = \left[\sum_{i \in S} k_s(l_i - l_0)\mathbf{u_i}\right] - k_d\dot{\mathbf{x}}, \tag{1.1}$$

kde $\ddot{\mathbf{x}} = \begin{bmatrix}\ddot{x}\\ \ddot{y}\end{bmatrix}$ je vektor vzájemného zrychlení konců pružiny, S je množina všech pružin jednoho robotu, k_s je tuhost pružiny, l_i je délka i-té pružiny, l_0 je délka volné pružiny, $\mathbf{u_i} = \begin{bmatrix}x\\ y\end{bmatrix}$ je jednotkový směrový vektor i-té pružiny, který určuje směr působení pružiny, k_d je tlumení působící na pružiny, $\dot{\mathbf{x}} = \begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}$ je vektor rychlosti mezi konci pružiny, m je hmotnost, která bude v tomto případě jednotková.

Leader following

Každý z robotů včetně vedoucího bude vybaven sadou optických a unikátních značek, které budou rozpoznávány kamerou. Takto je možné sledovat pozice

všech členů formace a určit tak vzdálenost a natočení mezi vedoucím a následujícím robotem. Díky známé relativní pozici ostatních robotů je možné mezi nimi navzájem vytvořit imaginární pružiny. K výpočtu rychlosti sledujícího robotu byla rovnice 1.1 upravena aplikováním předpokladu nulového relativního zrychlení do tvaru

$$\dot{\mathbf{x}} = \frac{[\sum_{i \in S} k_s(l_i - l_0)\mathbf{u_i}]}{k_d}. \tag{1.2}$$

Tuhost pružiny k_s je konstantní, proto rychlost sledujícího robotu roste lineárně se vzdáleností vedoucího robotu. Volná délka l_0 této pružiny je nastavena dle typu formace. Následně po dosazení vzdálenosti l_i, kterou získáme pomocí kamery a značek, je možné dopočítat rychlost ve směru obou os, kterou se musí sledující robot pohybovat.

1.2.3 Algoritmus potenciálových polí

Tento algoritmus využívá potenciálových polí k udržení požadovaného tvaru formace [6]. Důvodem pro jejich použití je jednoduchá reprezentace cíle a překážek. Pro generování těchto polí byla použita dvourozměrná funkce [5]

$$f(x, y) = e^{-\alpha(x-x_c)^2-\beta(y-y_c)^2}, \tag{1.3}$$

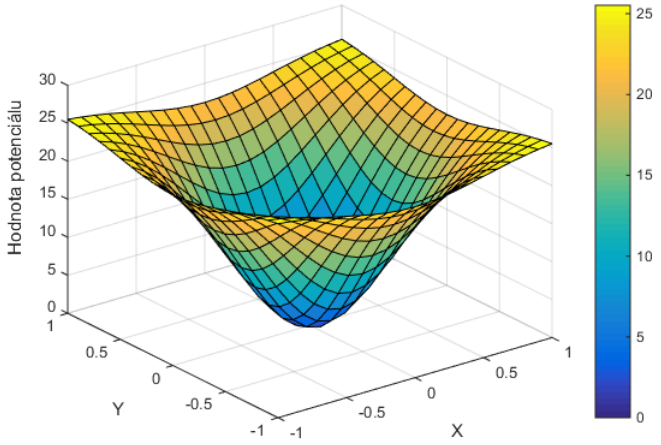
která má kruhovitý/eliptický charakter. Tato funkce generuje harmonické potenciálové pole, které se vyznačuje jedním globálním minimem a maximem, proto je tato funkce vhodná pro navádění robota prostředím, nehrozí totiž uvíznutí robota v lokálním minimu. Aktuální pozice robotu je dána souřadnicemi (x, y), střed funkce je reprezentován (x_c, y_c). Proměnné α a β určují eliptický tvar funkce.

Pro popis polohy robotu je použita pravotočivá kartézská souřadná soustava s počátkem umístěným ve středu příslušného robotu, přičemž daný robot se pohybuje v kladném směru osy y (obrázek 1.2).

Leader following

Funkce 1.3 generující potenciálová pole byla upravena tak, aby v centru funkce (x_c, y_c) byla hodnota potenciálu nulová

$$f_g(x, y) = K(1 - e^{-\alpha(x-x_{gc})^2-\beta(y-y_{gc})^2}). \tag{1.4}$$



Obrázek 1.4: Tvar funkce 1.4 s nulovou hodnotou potenciálu ve středu pro $\alpha = \beta = 2$ a $K = 26$. Potenciálová jáma reprezentuje ideální polohu robotu ve formaci.

Robot z údajů z kamery zjistí svoji pozici relativně k vedoucímu robotu. Pozice středu funkce 1.4 (x_c, y_c) je pevně určena vzhledem k pozici vedoucího robotu podle zvoleného typu formace. Robot se pak drží v potenciálové jámě, jejíž střed se pohybuje dle pohybu vedoucího robotu.

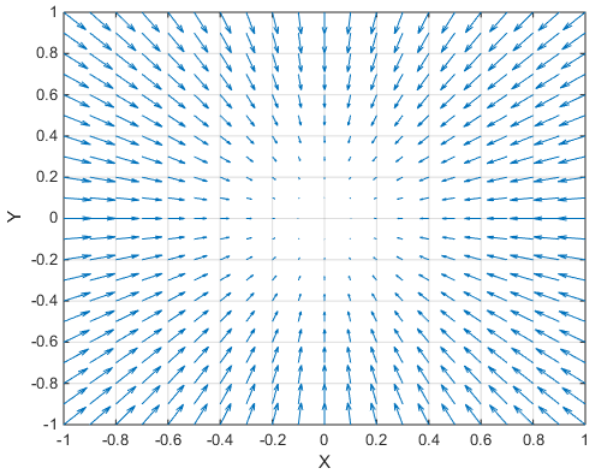
Parciálními derivacemi funkce 1.4 podle proměnných x a y lze získat vektor (d_{gx}, d_{gy}) [5], kde

$$d_{gx} = (f_g(x, y) - K)(-2\alpha(x - x_{gc})), \tag{1.5}$$

$$d_{gy} = (f_g(x, y) - K)(-2\beta(y - y_{gc})). \tag{1.6}$$

Vektor (d_{gx}, d_{gy}) je gradientem potenciálového pole (obrázky 1.4, 1.5). V každém bodě tedy určuje směr největšího klesání a vytváří tak vektorové pole, ve kterém každý vektor směřuje do středu potenciálové jámy (obrázek 1.5). Tento vektor normovaný svou velikostí poskytne jednotkový vektor směru rychlosti robotu ke své výchozí pozici ve formaci. Jednotkový vektor rychlosti je následně vynásoben hodnotou potenciálové funkce f_g v daném bodě, čímž je nastavena velikost výsledného vektoru rychlosti

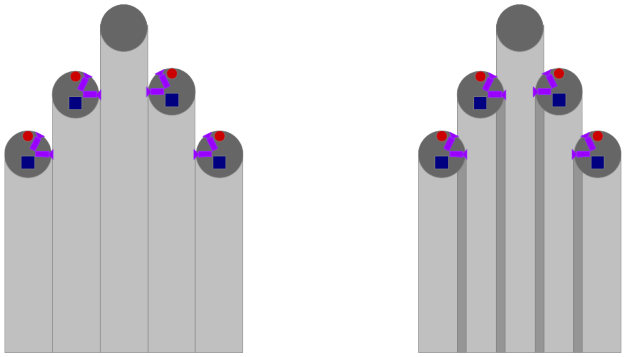
$$\dot{\mathbf{x}}_\mathbf{g} = \frac{\dot{\mathbf{x}}_\mathbf{g}}{|\dot{\mathbf{x}}_\mathbf{g}|} f_g. \tag{1.7}$$



Obrázek 1.5: Vektorové pole (d_{gx}, d_{gy}) pro potenciálovou jámu.

1.3 Rozbor formací

Tvarů formací existuje velké množství. K tomu, aby byla vybrána vhodná, je třeba vzít v úvahu několik faktorů. Prvním faktorem, který má vliv na tvar, je účel formace. Ta by pro náš účel měla být uspořádána takovým způsobem, aby jeden robot jel těsně vedle dráhy sousedního a ideálně neměly žádné překryvy ani mezery ve svých plochách jako na obrázku 1.6 a. Tím by byla zajištěna nejkratší doba nutná pro pokrytí celé plochy. Pro reálné řešení je ale vhodné počítat s nedokonalostí reálného řízení formace a ponechat malý, ale nezanedbatelný překryv (obrázek 1.6 b). Takový postup zamezí vzniku nepokrytých částí plochy.

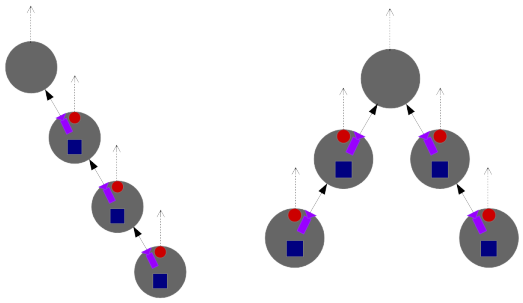


(a) : Ideální stopa robotické formace.

(b) : Stopa formace s překryvem.

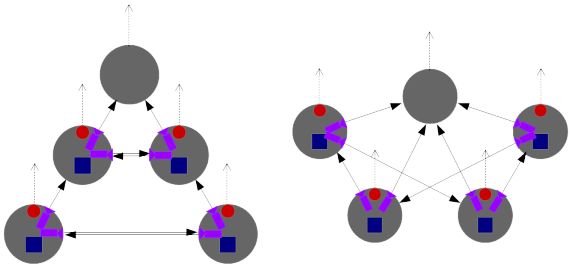
Obrázek 1.6: Stopa při udržování robotické formace pokrývající plochu.

Dalším kriteriem je množství dostupných senzorů polohy. V tomto případě byly použity kamery. Pokud by každý robot obsahoval pouze jedinou kameru s obvyklým úhlem záběru (60° - 80°), byly by možnosti struktury formace značně omezené, obrázek 1.7. Každý z robotů by tak stabilně mohl sledovat pouze jeden člen, který by pro něj byl vedoucím robotem. Snadno by se tak ze záběru kamery mohl tento vedoucí robot ztratit a v důsledku toho by se vazba uvnitř formace mohla přerušit a došlo by k odpojení robotu od formace včetně všech jeho následníků.



Obrázek 1.7: Struktury formace s využitím pouze jedné kamery.

Pokud by k dispozici byly dvě nebo více kamer, pak je možné každou natočit jiným směrem a sledovat tak více robotů současně. Tím by se zamezilo úplnému roztržení formace v případě ztráty robotu z obrazu z jedné z kamer. I kdyby se některá z vazeb na krátkou dobu přerušila, ostatní by měly být schopny daný robot udržet ve formaci.

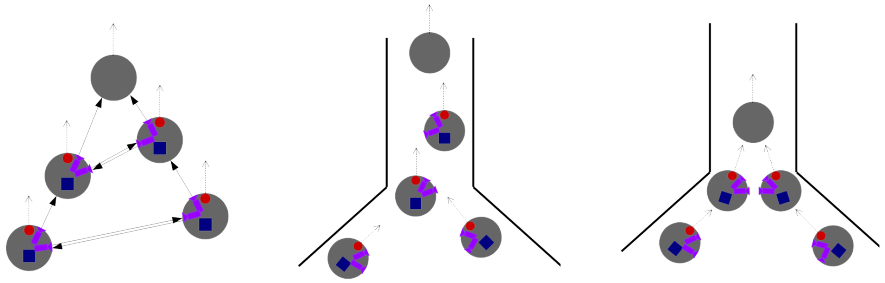


Obrázek 1.8: Struktura formace obsahující více vazeb mezi roboty.

V ideálním hejnu by každý robot měl mít informaci o poloze všech ostatních členů. To je reálně obtížné zajistit. Každý robot by tedy měl být vybaven alespoň dvěma kamerami, aby byl schopný sledovat více než jednoho člena formace (obrázek 1.8).

Dalším kritériem pro volbu vhodné formace může být prostředí, ve kterém se swarm bude pohybovat. Pokud se v oblasti budou vyskytovat zúžená místa, je třeba zvolit formaci, pro kterou bude jednodušší přizpůsobit tvar překážce

a následně se opět rozprostřít. Lze využít formace ukázané na obr. 1.8, pokud robotům, které jsou na stejné úrovni, posuneme ideální polohu vzhledem k vedoucímu robotu tak, aby se nenacházely v jedné rovině (obrázek 1.9 a). Pro tuto pozměněnou formaci bude jednodušší a rychlejší projet překážkou (obrázek 1.9 b). Mimo jiné, se tak dá předejít situaci, kdy by dva roboty vjely do zúženého místa současně (obrázek 1.9 c).



(a) : Upravená struktura formace.

(b) : Uspořádání během průjezdu.

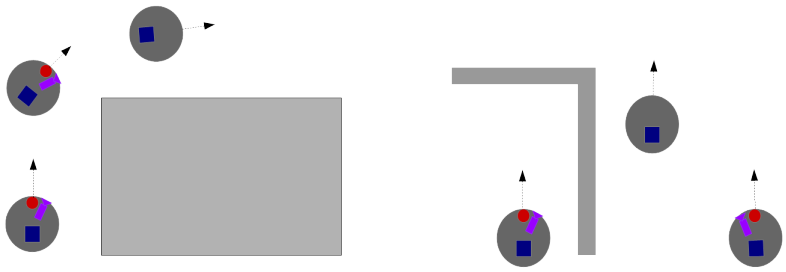
(c) : Příklad komplikace při průjezdu.

Obrázek 1.9: Příklad upravené formace pro průjezd zúženým místem, průběh průjezdu a možný chybový stav při průjezdu.

Vazby mezi roboty by se neměly příliš lišit v délce. Prakticky se vyskytující situace ukazují, že pokud budou roboty spojené vazbou příliš vzdáleny, budou obtížně měnit pozici ve formaci, například při stlačení během průjezdu zúženým místem, neboť budou muset překonat větší odpudivé síly virtuálních pružin.

1.4 Obnova po provozním selhání

Při reálném použití formace robotů je nutné vzít v úvahu případy, kdy se vedoucí robot ztratí z obrazu kamery. Takové situace mohou nastat velmi snadno. Jednou z takových situací může být případ, kdy se vedoucí robot ztratí za překážkou, jakou může být například sloup nebo roh zdi. V takovém případě je vhodné zajistit predikci pohybu vedoucího robotu. To lze dosáhnout užitím Kalmanova filtru, jehož princip je podrobněji rozebrán v kapitole 2.2.4. Predikci lze ovšem využít pouze pro krátký časový úsek (v řádu několika sekund), po této době se stane příliš nepřesnou. Pokud po uplynutí této doby robot stále nenajde svůj vedoucí robot, je možné dle aktuální situace vyzkoušet několik metod pro obnovení vizuálního kontaktu.



(a) : Průjezd formace okolo rohové překážky.

(b) : Zachycení robotu za překážkou tvaru L.

Obrázek 1.10: Příklady chybových stavů.

V případě, že se v záběru kamery nenachází vedoucí robot, ale nachází se zde jeden z ostatních následovníků (obrázek 1.10 a), je možné tento robot dočasně využít jako vedoucí, dokud se do záběru kamery opět nedostane správný robot.

Pokud se v záběru kamery nenachází žádný ze členů formace, je možné jako první pokus pro navázání vizuálního kontaktu zkusit otočení robotu kolem jeho středu tak, aby kamera postupně zachytila celý obraz prostoru (360°) okolo robotu. V případě, že kamerou zachytí některého ze členů formace, lze tento robot dočasně využít jako vedoucí.

Pokud by robot byl schopen zaznamenat příkazy, kterými se řídil, nebo dráhu, kterou dříve ujel (např. z odometrie), pak je možné předat robotu inverzní příkazy tak, aby se vrátil do bodu, ve kterém ještě dokázal vedoucí robot následovat. V tomto bodu, pokud to lze, se robot může napojit na svůj vedoucí robot, nebo vyzkoušet metody zde popsané. Této metody lze využít v situaci, která je ukázána na obrázku 1.10 b. V naznačené situaci by predikce ani hledání členů formace otáčením robotu nedokázaly nalézt vedoucí robot. V případě neúspěchu lze užít metodu náhodného prohledávání prostoru.

V případě, že je v rámci formace vytvořena komunikační síť, je možné prostřednictvím této sítě sdělit vedoucímu robotu informaci o ztrátě člena formace.

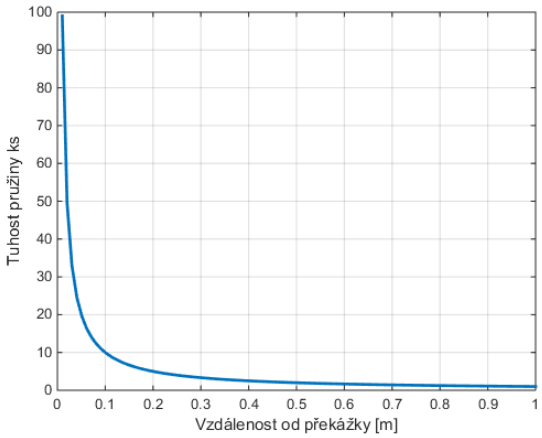
Kapitola 2

Realizace algoritmů

2.1 Collision avoidance

2.1.1 Pružiny

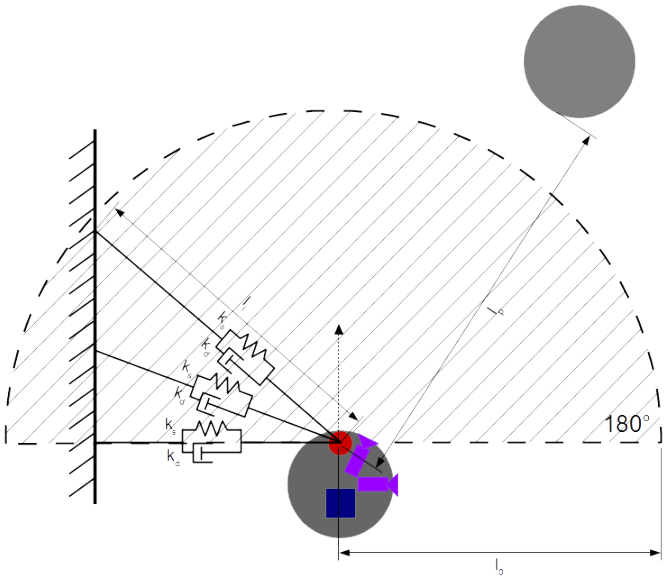
Pro detekci překážek byl použit laserový dálkoměr, který poskytuje měření přes celou horizontální polorovinu (úhel záběru 180°). Ve zde zvoleném řešení je každý z paprsků dálkoměru nahrazen imaginární pružinou podle obrázku 2.2. Hodnoty z dálkoměru se započítávají podmíněně, a jen tehdy pokud je



Obrázek 2.1: Průběh tuhosti pružiny při přibližování k překážce.

vzdálenost l_i menší než nastavená volná délka pružiny l_0. Tudíž je pružina vždy stlačována, a tedy robot pouze odpuzuje od překážky a nikdy naopak.

Tuhost pružin k_s není konstantní jako u pružin pro udržování formace, ale je reciprokou hodnotou vzdálenosti, což zajistí velkou sílu pružiny, pokud se vzdálenost bude limitně blížit nule. Překonání jakékoli síly z pružiny vytvořené sledováním vedoucího robotu je tak vždy zajištěno. Průběh tuhosti je zobrazen na obr. 2.1.



Obrázek 2.2: Nahrazení paprsků laserového dálkoměru imaginárními pružinami. Klidová délka pružiny l_0 určuje vzdálenost detekce překážek, proto překážka vzdálená l_p, kde $l_p > l_0$, nebude detekována. Záběr laserového dálkoměru je 180°.

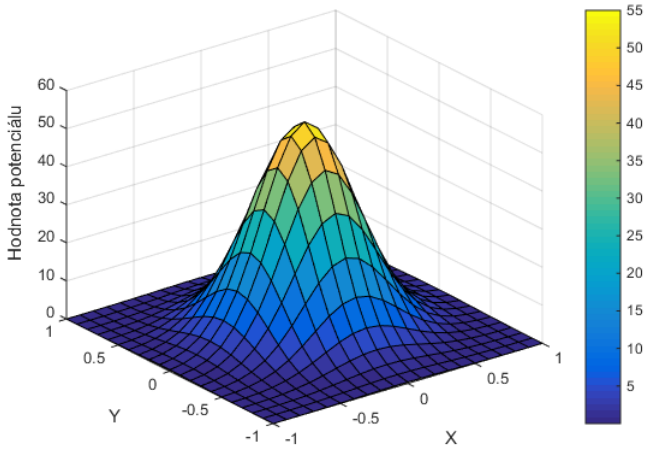
Pro výpočet je použita stejná souřadná soustava jako v kapitole 1.2.2 (obrázek 1.2). Z každé pružiny je za pomoci rovnice 1.2 vypočítán vektor rychlosti [24]. Všechny takto vytvořené vektory jsou sečteny do výslednice, která je podělena počtem započítaných vektorů. Tím se mimo jiné minimalizuje chyba, která by mohla být způsobena chybným měřením vzdálenosti. Na výslednici taková chyba bude mít malý vliv, pokud bude počítána z dostatečně velkého množství hodnot.

2.1.2 Potenciálová pole

Z každé změřené vzdálenosti, která je menší než stanovený limit pro započítání, a úhlu, pod kterým byla hodnota změřena, je vypočítána relativní poloha k danému robotu. V každém takto zjištěném bodě je umístěn střed (x_{oc}, y_{oc}) funkce [5]

$$f_o(x, y) = Ke^{-\alpha(x-x_{oc})^2-\beta(y-y_{oc})^2}, \tag{2.1}$$

jejíž průběh je zobrazen na obrázku 2.3. Parametr K slouží k nastavení maximální hodnoty potenciálu. Maximální hodnota potenciálové funkce by měla být mnohem vyšší než je hloubka potenciálové jámy, aby nedošlo k výraznému snížení maximální hodnoty potenciálu v místě překážky nebo dokonce k vyrušení obou polí, následkem čehož by robot nebyl odpuzován od překážky.



Obrázek 2.3: Tvar funkce (2.1) pro $\alpha = \beta = 5$ a $K = 55$.

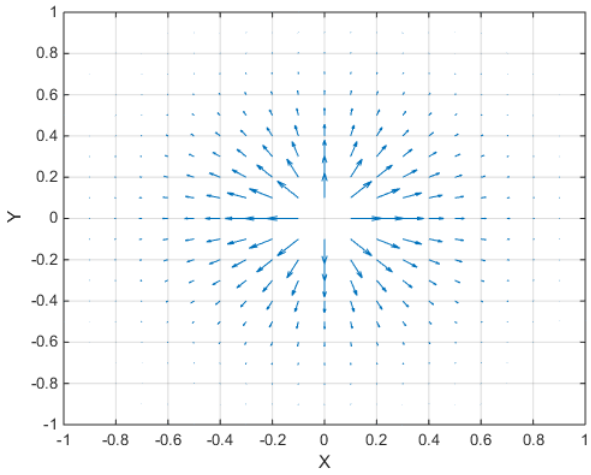
Střed funkce 2.1 je místo s nejvyšším potenciálem. Pro výpočet je použita stejná souřadná soustava jako v kapitole 1.2.3 (obrázek 1.2). Vektor $(-d_{ox}, -d_{oy})$ bude mít směr největšího klesání této funkce, kde

$$d_{ox} = f_o(x, y)(-2\alpha(x - x_{oc})), \tag{2.2}$$

$$d_{oy} = f_o(x, y)(-2\beta(y - y_{oc})). \tag{2.3}$$

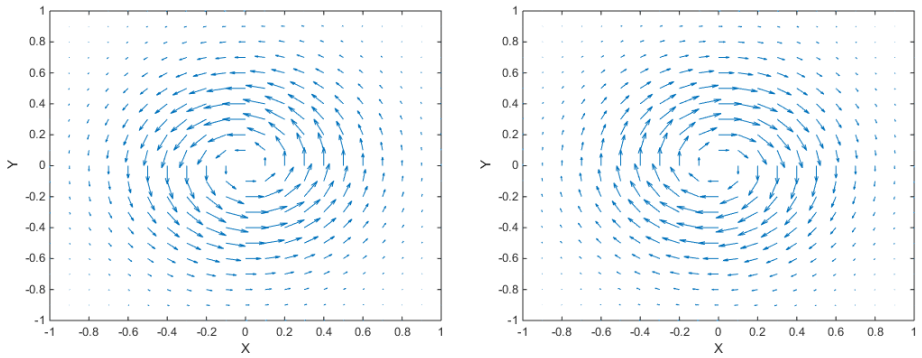
Gradientem funkce 2.1 je vytvořeno vektorové pole [5], [7], ve kterém každý vektor směřuje ze středu pole (obrázek 2.4) a robot je tak v každém místě odpuzován od překážky. Tento vektor normovaný svou velikostí poskytne jednotkový vektor směru rychlosti robotu ke své výchozí pozici ve formaci. Jednotkový vektor rychlosti je následně vynásoben hodnotou potenciálové funkce f_o v daném bodě, čímž je nastavena velikost výsledného vektoru rychlosti

$$\dot{\mathbf{x}}_\mathbf{o} = \frac{\dot{\mathbf{x}}_\mathbf{o}}{|\dot{\mathbf{x}}_\mathbf{o}|} f_o. \tag{2.4}$$



Obrázek 2.4: Vektorové pole $(-d_{ox}, -d_{oy})$ pro překážku.

Pro snazší objíždění překážek je možné využít ortogonálního vektorového pole vzhledem k poli z obrázku 2.4. Takové pole lze vytvořit otočením vektorů o $\pi/2$ rad nebo o $-\pi/2$ rad. Použití pouze ortogonálních vektorových polí z obr. 2.5 není vhodné. Při kolmé jízdě robotu k překážce by vektor výsledné rychlosti ležel pouze na ose x (dle obrázku 1.2). Výsledný vektor rychlosti by tedy obsahoval nulovou hodnotu složky y nebo hodnotu velmi blízkou nule. Takový vektor by při přiblížení nedokázal robot zpomalit a zajistit tak dostatek času na změnu směru, neboť čím vyšší rychlostí robot jede, tím větší setrvačná síla na něj působí a tím obtížněji mění směr jízdy. Snadno by se tak mohlo stát, že robot nestihne dostatečně a včas změnit směr jízdy a mohl by do překážky narazit.

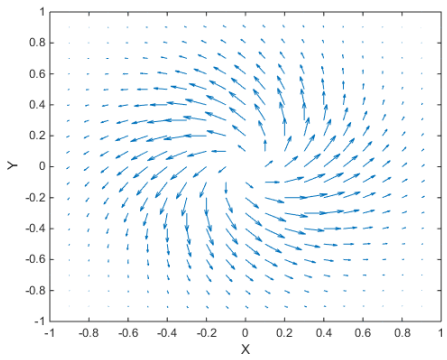


(a) : Ortogonální vektorové pole $(-d_{oy}, d_{ox})$ pro překážku.

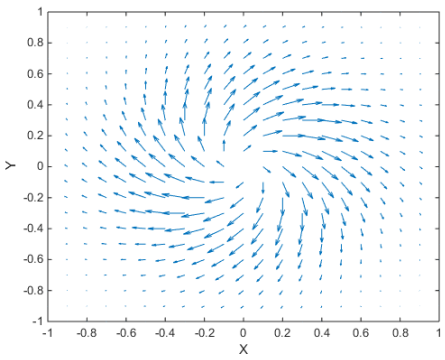
(b) : Ortogonální vektorové pole $(d_{oy}, -d_{ox})$ pro překážku.

Obrázek 2.5: Rotační vektorová pole pro objíždění překážky.

Vhodné výsledné vektorové pole, které předchozí nedostatek eliminuje, vznikne kombinací pole z obrázku 2.4 a jednoho z polí z obrázku 2.5. Takto složené pole obsahuje vektory složené jak ze složky, která robot při přiblížení přibrzdí, tak i ze složky, která robot nutí překážku objíždět.



(a) : Vektorové pole $(d_{ox} - d_{oy}, d_{oy} + d_{ox})$ pro překážku.



(b) : Vektorové pole $(d_{ox} + d_{oy}, d_{oy} - d_{ox})$ pro překážku.

Obrázek 2.6: Vírová vektorová pole pro překážku.

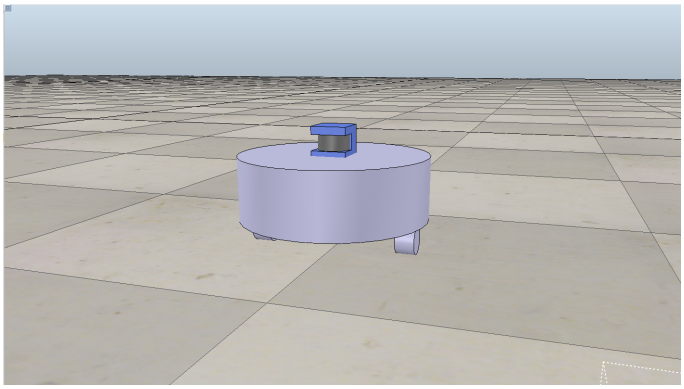
Typ vírového pole je zvolen podle umístění překážky relativně k danému robotu tak, aby robot, pokud je to možné, uhýbal směrem do formace. Tím bude zajištěna větší bezpečnost a kontrola nad řízením formace, neboť výhled kamery bude méně často blokován překážkou.

Optimální natočení druhé kamery by mělo být směrem do formace tak, aby mohla zachytit co nejvíce ostatních členů formace. Takto detekované roboty jsou považovány za překážky a pro jejich popis je použita funkce 2.1 a to i se stejnou hodnotou K, jako u obyčejné překážky. Liší se pouze v hodnotě α a β, které jsou voleny vyšší. To zajistí menší vliv na směr pohybu robotu pro velkou vzájemnou vzdálenost, ale velký vliv při přiblížení.

2.2 Implementace

2.2.1 Simulační prostředí

Pro ověření funkčnosti výše uvedených algoritmů bylo zvoleno simulační prostředí V-REP [1] (Virtual Robot Experimentation Platform). Jedná se o univerzální program vhodný pro multi-robotické systémy. Lze v něm modelovat prakticky jakýkoliv typ robotu a obsahuje také rozsáhlou nabídku modelů senzorů. Mimo jiné lze modelovat věrohodné prostředí pro testování. Program podporuje velké množství programovacích jazyků, jako jsou C/C++, Java, Matlab nebo Lua.

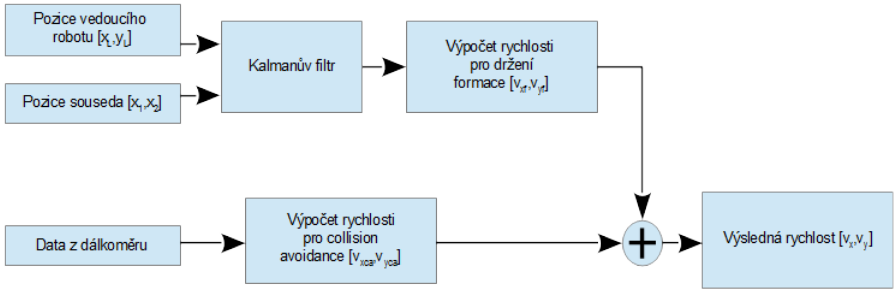


Obrázek 2.7: Model robotu s laserovým dálkoměrem použitým pro simulaci.

Byl vytvořen jednoduchý model robotu podobný reálnému exempláři systému, se kterým byl algoritmus později testován. V horní části měl připevněn laserový dálkoměr. Kamera pro detekci značek na ostatních robotech byla pro účely simulace nahrazena přesnou polohou převzatou ze simulace, ke které byl uměle přidán šum (cca $\pm 10\%$ naměřené vzdálenosti), což injektovalo chybu podstatně vyšší, než by byla chyba měření vzdálenosti z reálné kamery.

2.2.2 Řídící smyčka algoritmu

Hlavním vstupním parametrem pro řídící algoritmus jsou relativní souřadnice vedoucího robotu, na kterých závisí celé sledování. Dalším vstupem by měly být souřadnice ostatních robotů, které by měly podpořit předcházení kolizím robotů navzájem a zajistit celkově lepší stabilitu formace. Hodnoty souřadnic jsou dále upraveny pomocí Kalmanova filtru [2], jehož činnost je podrobněji vysvětlena v kapitole 2.2.4. Ten je schopen odstranit šum z naměřených hodnot, zmenšit případné mimořádné hodnoty (outliery) při chybných měřeních. Při ztrátě vstupních informací o poloze je schopen tuto polohu predikovat na základě předchozího vývoje dat. Pro výpočet rychlosti ze známé relativní polohy je použit jeden z výše uvedených algoritmů.



Obrázek 2.8: Diagram znázorňující princip výpočtu výsledné rychlosti robotu.

Dalším vstupním parametrem je vektor vzdáleností z dálkoměru. Tyto hodnoty jsou opět zpracovány jedním z výše uvedených algoritmů rozepsaných v kapitole 2.1, ze kterých dostaneme vektor pro vyhnutí překážce. Finální rychlost robotu je výslednicí z vektoru rychlosti pro sledování vedoucího robotu a pro vyhýbání se překážkám. Průběh výpočtu rychlosti je zobrazen v diagramu na obrázku 2.8.

2.2.3 Konkrétní implementace důležitých částí algoritmů

Pružinový algoritmus

Níže (Algorithm 1) je uveden pseudokód pro collision avoidance s využitím pružinového algoritmu. Ten představuje výpočetní cyklus pro jednu polohu robotu. Pro sledování vedoucího robotu a udržování formace je princip velmi podobný, proto je zde uvedena pouze tato část.

Vstupní proměnná *pts* v části *input* obsahuje data naměřená laserovým dálkoměrem (pro udržování formace jsou vstupními daty polohy ostatních robotů). V části *init* je nastavena maximální vzdálenost pro zahájení činnosti systému předcházení srážkám. Dále se zde načítá počet započtených pružin a je zde také definován poloměr robota. V sekci *loop* se provádí výpočet samotné rychlosti podle rovnice 1.2, která je již výstupní hodnotou.

Algorithm 1 Collision avoidance - spring algorithm

```
1:  procedure CASpr
2:  input:
3:      pts ← laser scanner data.
4:  init:
5:      defDistp ← limit distance for obstacles.
6:      sprNum ← number of created springs, initialize with 1.
7:  loop:
8:      for i = 0, i < length of pts, i++ do
9:          if pts[i] < defDistp then
10:             xp, yp ← compute relative coordinations of obstacles.
11:             uxp, uyp ← compute unit vector from xp, yp.
12:             ksp ← reciprocal distance pts[i].
13:             kdp ← damping ratio.
14:             vxp, vyp ← previous vpx, vpy sum with actuals.
15:             sprNum ← sprNum + 1.
16:     if sprNum > 1 then
17:         sprNum ← sprNum - 1.
18:     vxp, vyp ← vpx, vpy divided with sprNum.
19: output:
20:     return vxp, vyp.
```

Algoritmus potenciálových polí

Níže (Algorithm 2) je ukázán pseudokód pro udržení tvaru formace s využitím algoritmu potenciálových polí. Vstupem jsou relativní polohy vedoucího robotu a ostatních robotů. V části *init* jsou definovány parametry potenciálových funkcí. Dále je zde výchozí pozice robotu vzhledem k vedoucímu robotu, která je definována konkrétní formací. V části *enum* je nejprve dopočítána poloha středu potenciálové jámy vytvořené funkcí 1.4. Následně je dopočítána hodnota funkce. Dále jsou vypočítány hodnoty parciálních derivací potenciálové funkce a následně je za jejich pomoci vytvořen jednotkový směrový vektor. Nakonec je tento jednotkový vektor vynásoben hodnotou potenciálové funkce. Tím je získán vektor rychlosti pro následování.

Stejné kroky jako v předchozím odstavci jsou použity i pro výpočet vektoru rychlosti pro interakci s okolními roboty. Místo potenciálové jámy je použita funkce 2.1. Její parametry α a β by měly být voleny vyšší než u klasické překážky, aby měla funkce malou hodnotu pro větší vzdálenosti mezi roboty a neovlivňovala je, pokud jsou na svých pozicích, ale aby měla velký vliv na pohyb pro malé vzdálenosti.

Stejný postup, jako je uvedený výše (Algorithm 2), lze využít i v případě, kdy se leader ztratí z obrazu kamery, ale v záběru kamery je jiný robot. Pak lze tento robot nouzově využít jako vedoucí robot, aby nedošlo k úplnému roztžení formace.

Algorithm 2 Formation - potential fields

```
1:  procedure FORMPF
2:  input:
3:      xL, yL ← relative coordinations of leader.
4:      xi, yi ← relative coordinations of any other robot.
5:  init:
6:      ag, bg, ai, bi ← set exponent parameters of functions.
7:      kg, ki ← set gain of functions.
8:      defxc, defyc ← default relative position to leader in formation.
9:  enum:
10:     xc, yc ← center of goal function, xL, yL + defxc, defyc.
11:     fg ← value of potential function for goal.
12:     fi ← value of potential function for other robots.
13:     dfgx, dfgy ← values of partial derivations of fg.
14:     dfix, dfiy ← values of partial derivations of fi.
15:     ufgx, ufgy ← unit vectors from dfgx, dfgy.
16:     ufix, ufiy ← unit vectors from dfix, dfiy.
17:     vfgx, vfgy ← vector ufg multiplied by value fg.
18:     vfix, vfiy ← vector ufi multiplied by value fi.
19: output:
20:     return vfgx, vfgy, vfix, vfiy.
```

Níže (Algorithm 3) je ukázán pseudokód vytváření funkcionality vyhýbání se překážkám s využitím algoritmu potenciálových polí. Vstupní proměnná *pts* obsahuje data z dálkoměru. V části *init* jsou nastaveny vzdálenost pro měření překážek, počet započítaných vzdáleností a parametry potenciálové funkce. Na řádcích 12 - 16 je počítán vektor rychlosti stejným principem jako v předchozí ukázce. Oblast okolo robotu je rozdělena do sekcí, kde první sekce je v blízkém okolí robotu a dochází v ní pouze k odpuzení, stejně tak jako v části přímo před robotem, podle obrázku 2.4. V těchto případech je důležitější robot zpomalit, či zastavit, aby měl možnost změnit směr jízdy. Ve dvou zbývajících sekcích na stranách robotu jsou vytvořena vírová odpudivá pole, která jsou zobrazena na obrázku 2.6, pro snadnější objíždění překážek.

Algorithm 3 Collision avoidance - potential fields algorithm

```
1:  procedure CAPF
2:  input:
3:      pts ← laser scanner data.
4:  init:
5:      defDistp ← limit distance for obstacles.
6:      beamNum ← number of created springs, initialize with 1.
7:      ao, bo ← set exponent parameters of function.
8:      ko ← set gain of function.
9:  loop:
10:     for i = 0, i < length of pts, i++ do
11:         if pts[i] < defDistp then
12:             xp, yp ← compute relative coordinations of obstacles.
13:             fo ← value of potential function for obstacles.
14:             dfox, dfoy ← values of partial derivations of fo.
15:             ufox, ufoy ← unit vectors from dfox, dfoy.
16:             vfox, vfoy ← vector ufo multiplied with value fo.
17:             if pts[i] < 0.3 then
18:                 vxp ← vxp + vfox.
19:                 vyp ← vyp + vfoy.
20:             else if angle[i] < 70° then
21:                 vxp ← vxp + vfox + vfoy.
22:                 vyp ← vyp + vfoy - vfox.
23:             else if angle[i] < 110° then
24:                 vxp ← vxp + vfox.
25:                 vyp ← vyp + vfoy.
26:             else
27:                 vxp ← vxp + vfox - vfoy.
28:                 vyp ← vyp + vfoy + vfox.
29:             beamNum ← beamNum + 1.
30:     if beamNum > 1 then
31:         beamNum ← beamNum - 1.
32:     vxp, vyp ← vpx, vpy divided by beamNum.
33: output:
34:     return vxp, vyp.
```

2.2.4 Kalmanův filtr

Kalmanův filtr je matematický aparát [25], který slouží pro filtraci zašuměných signálů v časové oblasti bez nutné znalosti parametrů rušení. Jedná se o predikční algoritmus schopný předpovědět průběh budoucích hodnot signálů z průběhu hodnot minulých.

Výhodou tohoto filtru je, že dokáže pracovat v časové oblasti, oproti jiným, které pracují ve frekvenční oblasti. Odpadá tak implementačně i výpočetně náročná Fourierova transformace. Dále vzhledem k jiným filtračním algoritmům, které pracují v časové oblasti, není nutné znát předem parametry hledaného signálu ani rušení.

Pro výpočet je předpokládán stavový model složený ze dvou částí [2]. Z modelu dynamiky stavu, který říká jak závisí příští stav na předchozím stavu

$$x_n = Ax_{n-1} + Bu_n + w_n, \tag{2.5}$$

kde x reprezentuje stavy, u jsou vstupy modelu, w je bílý šum a A a B jsou stavové matice. Druhou částí je model měření výstupu, který říká, jak závisí aktuální výstup na aktuálním stavu

$$y_n = Cx_n + Du_n + v_n, \tag{2.6}$$

kde y je aktuální hodnota výstupu, v je bílý šum a C a D jsou stavové matice.

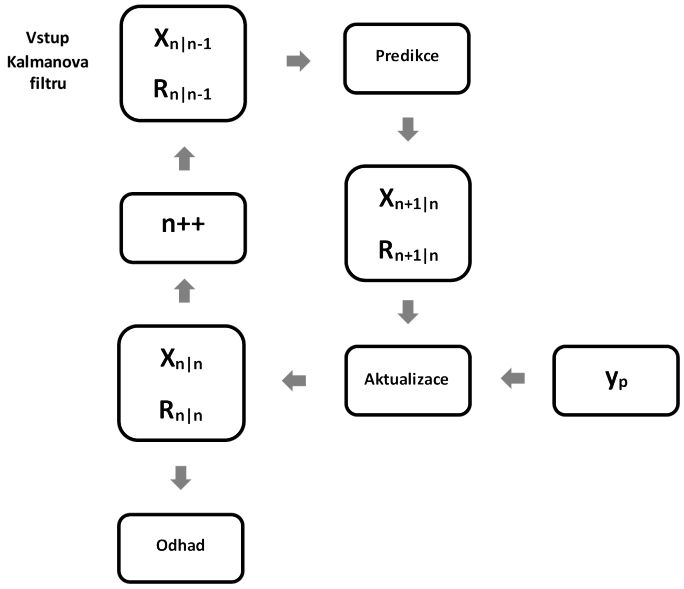
Kalmanův filtr se skládá ze dvou částí, z filtrace a predikce [2].

Výstupem filtrace je odhad pro střední hodnoty položek stavového vektoru. Ten bude značen $x_{n|n}$, kde první n znamená odhad stavu pro n-tý cyklus a druhé n znamená výpočet z n-tých dat.

Druhým výstupem filtrace je kovarianční matice pro prvky stavového vektoru, která je značena $R_{n|n}$. Prvky na diagonále kovarianční matice určují rozptyly pro jednotlivé prvky stavového vektoru. Směrodatné odchylky z nich vytvořené pak určují, jak bude sdružená Gaussova křivka v určitém směru široká. Čím menší tyto rozptyly jsou, tím přesněji budou prvky stavového vektoru určeny. Členy mimo diagonálu určují závislosti mezi stavovými proměnnými.

Druhou částí Kalmanova filtru je predikce, jejímž výstupem jsou odhady pro střední hodnoty stavových proměnných a kovarianční matice pro další cyklus. Vektor odhadů středních hodnot bude značen $x_{n+1|n}$ a kovarianční matice pro prvky stavového vektoru $R_{n+1|n}$.

Vstupem Kalmanova filtru jsou odhady středních hodnot stavových proměnných $x_{n|n-1}$ a kovarianční matice $R_{n|n-1}$ z předchozího cyklu.



Obrázek 2.9: Diagram Kalmanova filtru. Převzato z [14] 23.5.2017.

Rovnice pro algoritmus Kalmanova filtru [2] jsou následující:

- předpověď výstupu
$$y_p = Cx_{n|n-1} + Du_n, \tag{2.7}$$
- kovariance výstupu
$$R_p = r_v + CR_{n|n-1}C', \tag{2.8}$$
- přepočet kovariance stavu
$$R_{n|n} = R_{n|n-1} - R_{n|n-1}C'R_p^{-1}CR_{n|n-1}, \tag{2.9}$$
- Kalmanův zisk
$$K = R_{n|n}C'r_v^{-1}, \tag{2.10}$$
- datová oprava stavu
$$x_{n|n} = x_{n|n-1} + K(y_n - y_p), \tag{2.11}$$
- předpověď stavu
$$x_{n+1|n} = Ax_{n|n} + Bu_n, \tag{2.12}$$
- předpověď kovariance stavu
$$R_{n+1|n} = r_w + AR_{n|n}A'. \tag{2.13}$$

Kapitola 3

Testování stability formace v simulátoru

Na stabilitu formace lze pohlížet dvěma způsoby. Prvním je dynamická stabilita, která se zabývá stabilitou systému z hlediska regulace stavových proměnných při řízení robotů. Toto téma, které je velmi rozsáhlé, není předmětem této práce, proto byly parametry formace experimentálně zvoleny tak, aby systém v experimentech vykazoval aperiodické chování.

Druhým pohledem je kinematická stabilita, která se zabývá udržováním formace jako celku. V této kapitole budou z hlediska této stability testovány oba výše implementované algoritmy dle různých kritérií:

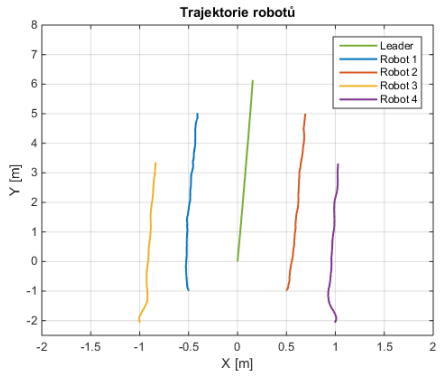
- Maximální vzdálenost robotu od výchozí pozice ve formaci.
- Doba ustálení odchylky vzdálenosti.
- Minimální vzdálenost od překážky.

Budou testovány pro dva různé typy formací, které jsou zobrazeny na obrázku 1.8. Bude také testována schopnost vyhnutí se překážkám a projetí zúžených míst.

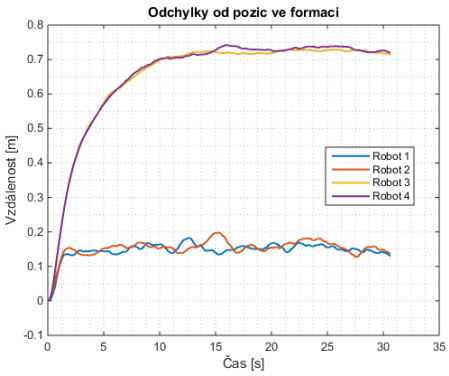
3.1 Testování kinematické stability formace

3.1.1 Jízda podél lineární trajektorie

Prvním experimentem je jízda po přímce s konstantní rychlostí vedoucího robotu, která testuje základní schopnost držení formace za jízdy a stanoví pozice robotů, na kterých se mají ustálit po průjezdu kolem překážky ve 3. a 4. pokusu.



(a) : Trajektorie robotů pro celkový čas jízdy $t = 30.6\,\mathrm{s}$.

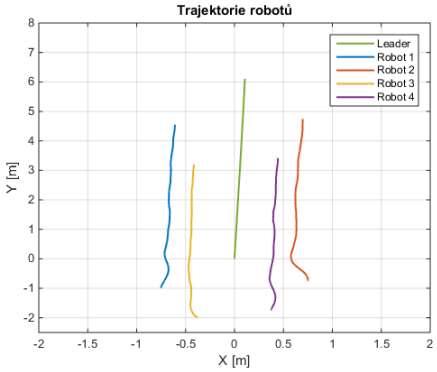


(b) : Průběh vzdáleností od výchozích poloh.

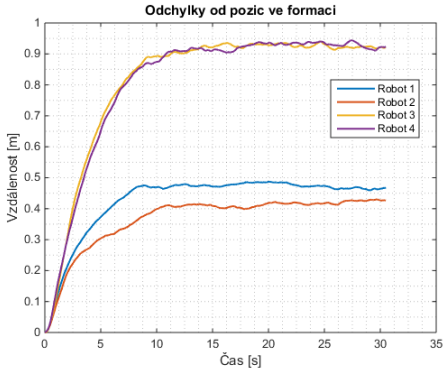
Obrázek 3.1: Průběh rovné jízdy pro pružinový algoritmus pro formaci ve tvaru šipky s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.5,-1.25], Robot 2 [0.5,-1], Robot 3 [-1,-2.25], Robot 4 [1,-2].

Charakteristické hodnoty - pružiny - šipka				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.18	0.15	0.15	0.15
2	0.20	0.16	0.16	0.15
3	0.73	0.72	0.72	0.65
4	0.74	0.73	0.72	0.65

Tabulka 3.1: Charakteristické hodnoty pro rovnou jízdu pro pružinový algoritmus pro formaci ve tvaru šipky.



(a) : Trajektorie robotů pro celkový čas jízdy $t = 30.5\,\mathrm{s}$.

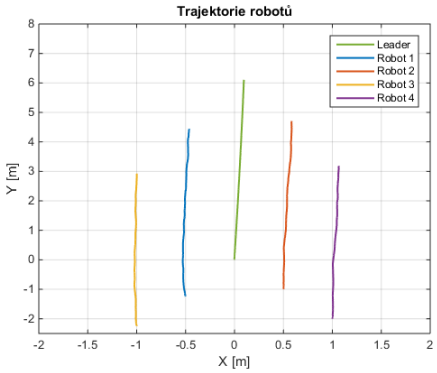


(b) : Průběh vzdáleností od výchozích poloh.

Obrázek 3.2: Průběh rovné jízdy pro pružinový algoritmus pro formaci ve tvaru W s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.75,-1], Robot 2 [0.75,-0.75], Robot 3 [-0.375,-2], Robot 4 [0.375,-1.75].

Charakteristické hodnoty - pružiny - W				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.20	0.16	0.16	0.16
2	0.20	0.11	0.12	0.11
3	0.47	0.42	0.43	0.41
4	0.53	0.47	0.47	0.44

Tabulka 3.2: Charakteristické hodnoty pro rovnou jízdu pro pružinový algoritmus pro formaci ve tvaru W.



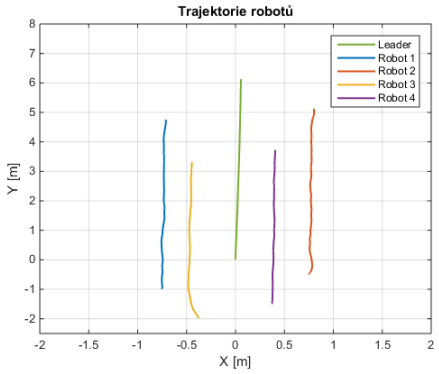
(a) : Trajektorie robotů pro celkový čas jízdy $t = 30.5\,\mathrm{s}$.

(b) : Průběh vzdáleností od výchozích poloh.

Obrázek 3.3: Průběh rovné jízdy pro algoritmus potenciálových polí pro formaci ve tvaru šipky s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.5,-1.25], Robot 2 [0.5,-1], Robot 3 [-1,-2.25], Robot 4 [1,-2].

Charakteristické hodnoty - pot. pole - šipka				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.50	0.43	0.44	0.42
2	0.51	0.41	0.41	0.40
3	0.99	0.90	0.87	0.80
4	0.91	0.83	0.83	0.77

Tabulka 3.3: Charakteristické hodnoty pro rovnou jízdu pro algoritmus potenciálových polí pro formaci ve tvaru šipky.



(a) : Trajektorie robotů pro celkový čas jízdy $t = 30.6\,\mathrm{s}$.

(b) : Průběh vzdáleností od výchozích poloh.

Obrázek 3.4: Průběh rovné jízdy pro algoritmus potenciálových polí pro formaci ve tvaru W s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.75,-1], Robot 2 [0.75,-0.5], Robot 3 [-0.375,-2], Robot 4 [0.375,-1.5].

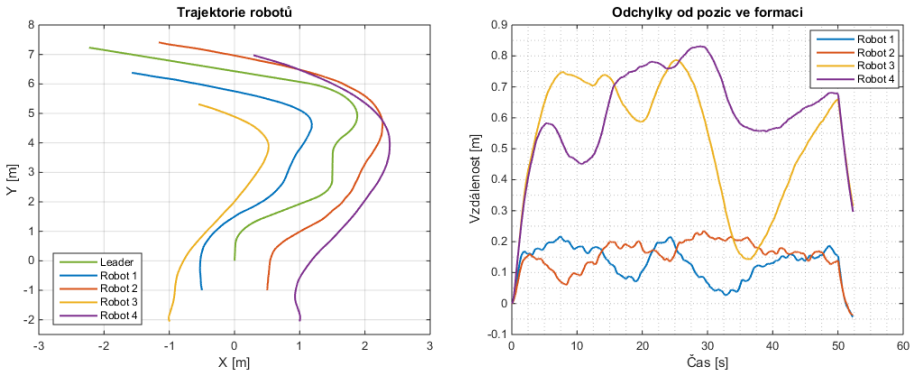
Charakteristické hodnoty - pot. pole - W				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.43	0.36	0.36	0.34
2	0.35	0.28	0.30	0.28
3	0.94	0.88	0.83	0.75
4	0.94	0.83	0.83	0.77

Tabulka 3.4: Charakteristické hodnoty pro rovnou jízdu pro algoritmus potenciálových polí pro formaci ve tvaru W.

Z obrázků 3.1 - 3.4 je vidět, že odchylka od polohy je větší pro roboty, které nejsou závislé přímo na vedoucím robotu, ale na jiném robotu. Každý robot reaguje s určitým zpožděním, které se v dalších úrovních formace kumuluje. Dále je vidět, že roboty řízené pružinovým algoritmem hůře sledují požadovanou trajektorii, ale mají menší zpoždění. U potenciálových polí je situace spíše opačná.

3.1.2 Jízda po křivce

Druhým experimentem je jízda po křivce, která by měla otestovat držení formace pro obtížnější trajektorie než je přímka. Použitá trajektorie vedoucího robotu je vidět na obrázcích 3.5 - 3.8 a je vyznačena zelenou barvou. Obsahuje například mírné zatáčky pro ověření citlivosti reakce sledujících robotů na menší změny v trajektorii vedoucího robotu.



(a) : Trajektorie robotů pro celkový čas jízdy $t = 52.3\,\mathrm{s}$.

(b) : Průběh vzdáleností od výchozích poloh.

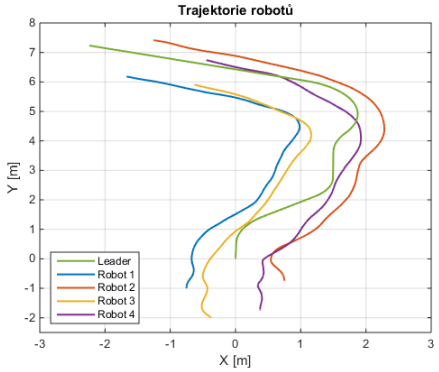
Obrázek 3.5: Průběh jízdy po křivce pro pružinový algoritmus pro formaci ve tvaru šipky s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.5,-1.25], Robot 2 [0.5,-1], Robot 3 [-1,-2.25], Robot 4 [1,-2].

Charakteristické hodnoty - pružiny - šipka				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.22	0.18	0.14	0.13
2	0.23	0.16	0.16	0.16
3	0.79	0.74	0.60	0.53
4	0.83	0.56	0.61	0.62

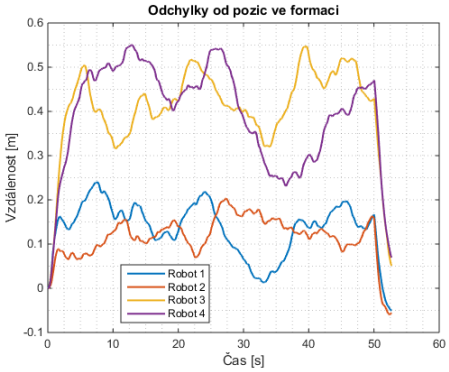
Tabulka 3.5: Charakteristické hodnoty pro jízdu po křivce pro pružinový algoritmus pro formaci ve tvaru šipky.

Charakteristické hodnoty - pružiny - W				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.24	0.15	0.15	0.13
2	0.20	0.13	0.12	0.12
3	0.55	0.40	0.42	0.41
4	0.55	0.45	0.43	0.39

Tabulka 3.6: Charakteristické hodnoty pro jízdu po křivce pro pružinový algoritmus pro formaci ve tvaru W.

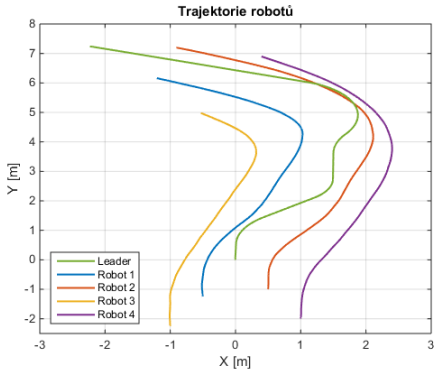


(a) : Trajektorie robotů pro celkový čas jízdy $t = 52.7\,\mathrm{s}$.

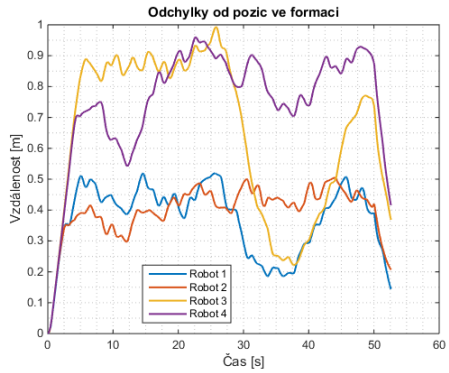


(b) : Průběh vzdáleností od výchozích poloh.

Obrázek 3.6: Průběh jízdy po křivce pro pružinový algoritmus pro formaci ve tvaru W s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.75,-1], Robot 2 [0.75,-0.75], Robot 3 [-0.375,-2], Robot 4 [0.375,-1.75].



(a) : Trajektorie robotů pro celkový čas jízdy $t = 52.6\,\mathrm{s}$.

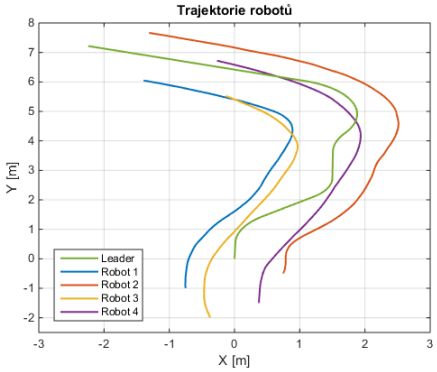


(b) : Průběh vzdáleností od výchozích poloh.

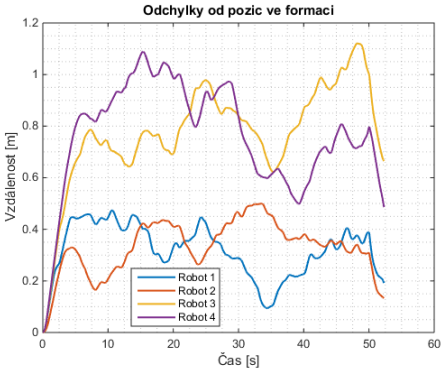
Obrázek 3.7: Průběh jízdy po křivce pro algoritmus potenciálových polí pro formaci ve tvaru šipky s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.5,-1.25], Robot 2 [0.5,-1], Robot 3 [-1,-2.25], Robot 4 [1,-2].

Charakteristické hodnoty - pot. pole - šipka				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.52	0.44	0.41	0.37
2	0.51	0.44	0.41	0.40
3	0.99	0.87	0.76	0.65
4	0.96	0.89	0.80	0.75

Tabulka 3.7: Charakteristické hodnoty pro jízdu po křivce pro algoritmus potenciálových polí pro formaci ve tvaru šipky.



(a) : Trajektorie robotů pro celkový čas jízdy $t = 52.3\,\mathrm{s}$.



(b) : Průběh vzdáleností od výchozích poloh.

Obrázek 3.8: Průběh jízdy po křivce pro algoritmus potenciálových polí pro formaci ve tvaru W s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.75,-1], Robot 2 [0.75,-0.5], Robot 3 [-0.375,-2], Robot 4 [0.375,-1.5].

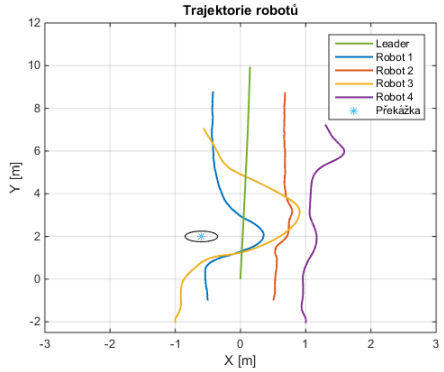
Charakteristické hodnoty - pot. pole - W				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.47	0.44	0.34	0.32
2	0.50	0.36	0.34	0.33
3	1.12	0.78	0.78	0.78
4	1.09	0.72	0.79	0.77

Tabulka 3.8: Charakteristické hodnoty pro jízdu po křivce pro algoritmus potenciálových polí pro formaci ve tvaru W.

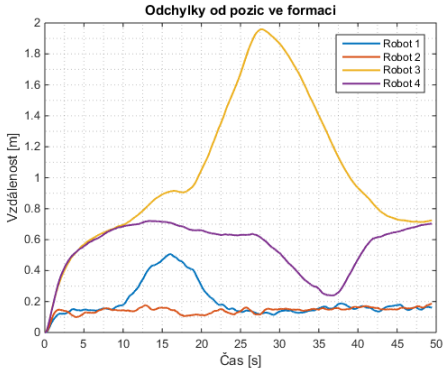
U pružinového algoritmu je vidět trochu větší citlivost na menší změny v jízdě vedoucího robota oproti algoritmu potenciálových polí. Avšak v obou případech je vidět, že citlivost na malé změny je nízká již u robotů, které přímo sledují vedoucí robot. U dalších robotů jsou reakce prakticky nulové. Dále je také vidět, že ve všech případech dochází u robotů ke zkracování oblouků, po kterých by se měly pohybovat. To je způsobeno tím, že algoritmy nepočítají se zpožděnou polohou, kterou mají vzhledem k vedoucímu robotu, a reagují na její změnu okamžitě.

3.1.3 Objíždění kompaktní překážky malého rozměru

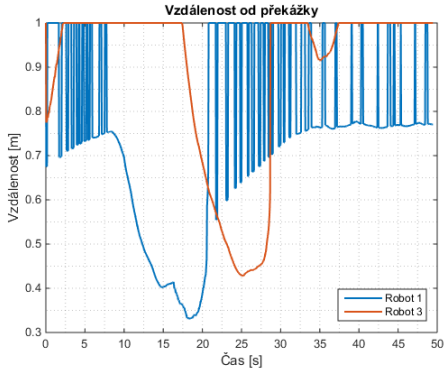
Třetím experimentem je rovná jízda, při které bude testována schopnost robotů vyhnout se překážce. Její poloha byla zvolena tak, aby co nejvíce zasahovala do trajektorie robotu. Jako překážka byl zvolen kruhový sloup s průměrem 0.5 m a s polohou [-0.6, 2] vzhledem k počáteční pozici vedoucího robotu.



(a) : Trajektorie robotů pro celkový čas jízdy $t = 49.5\,\mathrm{s}$.



(b) : Průběh vzdáleností od výchozích poloh.



(c) : Průběh nejmenších vzdáleností od detekovaných překážek.

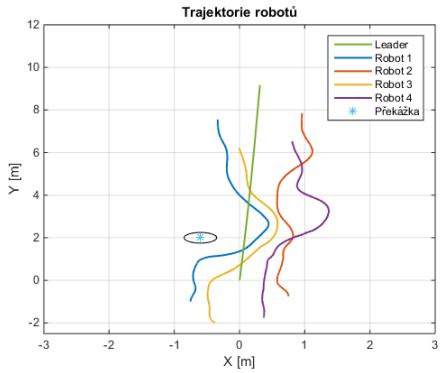
Obrázek 3.9: Průběh jízdy s objezdem překážky pro pružinový algoritmus pro formaci ve tvaru šipky s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.5,-1.25], Robot 2 [0.5,-1], Robot 3 [-1,-2.25], Robot 4 [1,-2].

Charakteristické hodnoty - pružiny - šipka				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.51	0.14	0.16	0.20
2	0.19	0.15	0.15	0.14
3	1.96	0.72	0.91	1.03
4	0.72	0.63	0.63	0.56

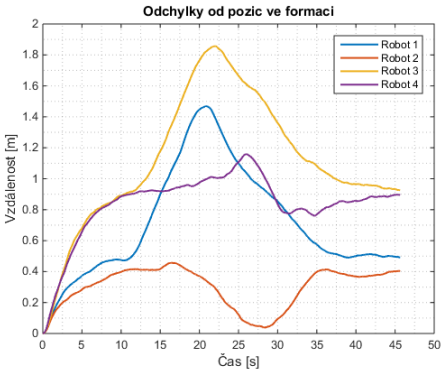
Tabulka 3.9: Charakteristické hodnoty pro jízdu okolo překážky pro pružinový algoritmus pro formaci ve tvaru šipky.

Z obrázku 3.9 je vidět, že členy formace se úspěšně vyhnuly překážce. První robot se ke sloupu přiblížil cca na 0.4 m, následně laserový dálkoměr detekoval robot 2 jako překážku a přiblížil se k němu asi na 0.33 m. Kmitavý charakter dat z laseru je způsoben malou hustotou paprsků dálkoměru a malou detekční oblastí na ostatních robotech. Robot 3 se přiblížil k překážce asi na 0.43 m, ovšem zařazení zpět do formace na očekávanou pozici trvalo dlouhou dobu

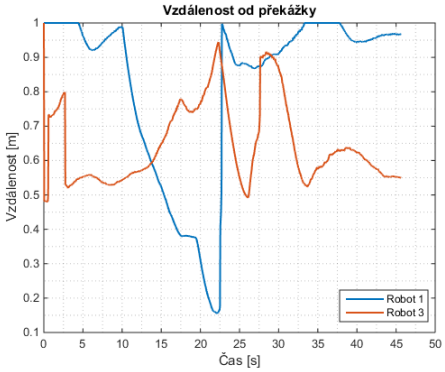
(cca až o 10 s déle). Dále je vidět, že překážka nezanedbatelně ovlivňuje i trajektorie robotů, v jejichž dráze se nenachází.



(a) : Trajektorie robotů pro celkový čas jízdy $t = 45.7\,\mathrm{s}$.



(b) : Průběh vzdáleností od výchozích poloh.



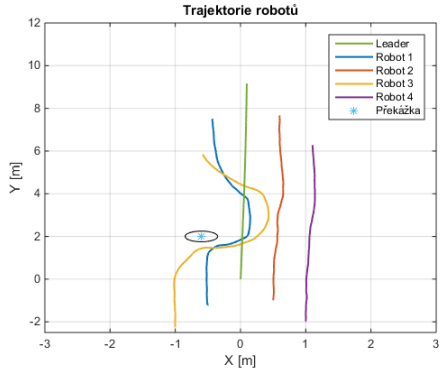
(c) : Průběh nejmenších vzdáleností od detekovaných překážek.

Obrázek 3.10: Průběh jízdy s objezdem překážky pro pružinový algoritmus pro formaci ve tvaru W s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.75,-1], Robot 2 [0.75,-0.75], Robot 3 [-0.375,-2], Robot 4 [0.375,-1.75].

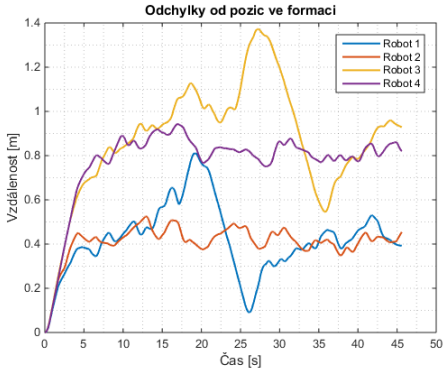
Charakteristické hodnoty - pružiny - W				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	1.47	0.50	0.57	0.72
2	0.46	0.41	0.37	0.30
3	1.86	0.96	1.03	1.12
4	1.16	0.92	0.88	0.84

Tabulka 3.10: Charakteristické hodnoty pro jízdu okolo překážky pro pružinový algoritmus pro formaci ve tvaru W.

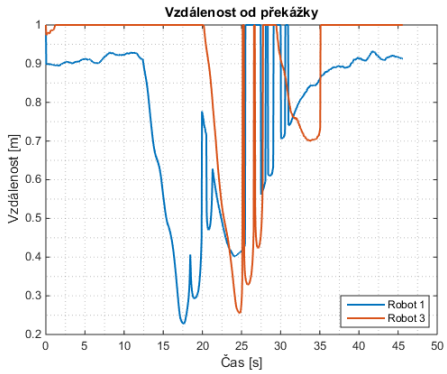
Z obrázku 3.10 je vidět, že členové formace se opět úspěšně vyhnuli překážce. Průběh vyhnutí byl podobný jako v předchozím případě. I přes vyšší hodnotu odchylky robotu 1 od výchozí pozice dosáhla formace ustálení asi o 10 s rychleji. Roboty, v jejichž dráze se překážka nenacházela, byly ovlivněny více než v předchozím případě.



(a) : Trajektorie robotů pro celkový čas jízdy $t = 45.6\,\mathrm{s}$.



(b) : Průběh vzdáleností od výchozích poloh.



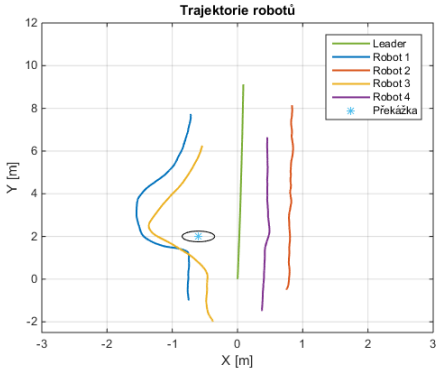
(c) : Průběh nejmenších vzdáleností od detekovaných překážek.

Obrázek 3.11: Průběh jízdy s objezdem překážky pro algoritmus potenciálových polí pro formaci ve tvaru šipky s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.5,-1.25], Robot 2 [0.5,-1], Robot 3 [-1,-2.25], Robot 4 [1,-2].

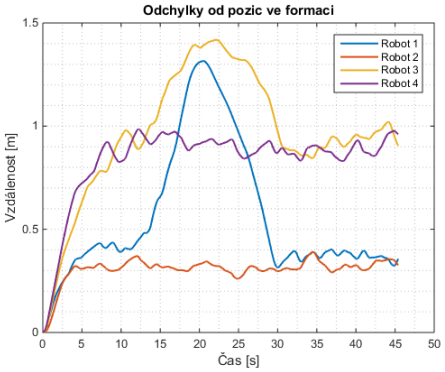
Charakteristické hodnoty - pot. pole - šipka				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.81	0.38	0.42	0.42
2	0.53	0.41	0.42	0.41
3	1.37	0.93	0.92	0.87
4	0.94	0.83	0.82	0.78

Tabulka 3.11: Charakteristické hodnoty pro jízdu okolo překážky pro algoritmus potenciálových polí pro formaci ve tvaru šipky.

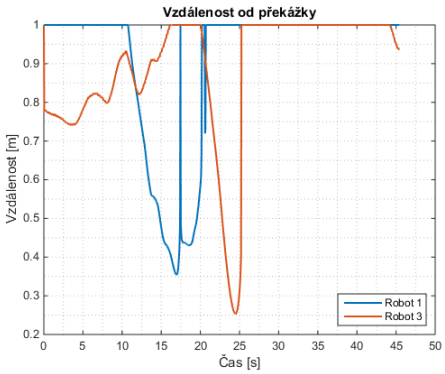
Z obrázku 3.11 je vidět, že se členové formace opět úspěšně vyhnuly překážce. Odchylka od výchozí polohy ve formaci byla nižší než v předchozích dvou případech cca o 0.5 m. K ustálení došlo zhruba ve stejném čase jako v předchozím případě, tedy kolem 35 s. Dále je také vidět, že roboty, které v dráze překážku nemají, nejsou překážkou téměř ovlivněné.



(a) : Trajektorie robotů pro celkový čas jízdy $t = 49.4\,\mathrm{s}$.



(b) : Průběh vzdáleností od výchozích poloh.



(c) : Průběh nejmenších vzdáleností od detekovaných překážek.

Obrázek 3.12: Průběh jízdy s objezdem překážky pro algoritmus potenciálových polí pro formaci ve tvaru W s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.75,-1], Robot 2 [0.75,-0.5], Robot 3 [-0.375,-2], Robot 4 [0.375,-1.5].

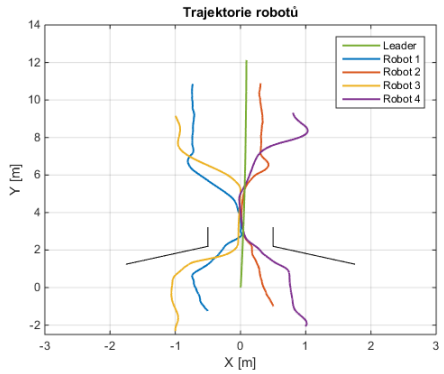
Charakteristické hodnoty - pot. pole - W				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	1.32	0.36	0.40	0.56
2	0.39	0.31	0.31	0.31
3	1.42	0.94	0.95	0.97
4	0.99	0.91	0.89	0.84

Tabulka 3.12: Charakteristické hodnoty pro jízdu okolo překážky pro algoritmus potenciálových polí pro formaci ve tvaru W.

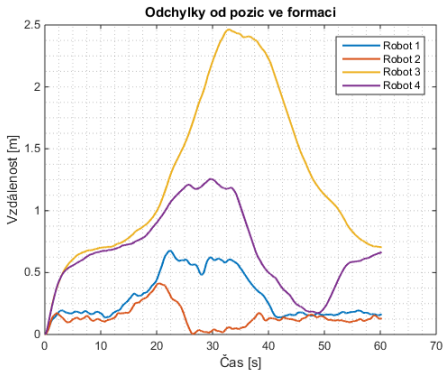
Z obrázku 3.12 je patrné, že se roboty opět překážce vyhnuly. Robot 1 dosáhl velké odchylky od výchozí pozice ve formaci. Přesto byl čas ustálení na pozicích kratší než v předchozích případech, cca 30 s. Zbylé roboty nebyly nijak překážkou ovlivněny.

3.1.4 Průjezd úžinou

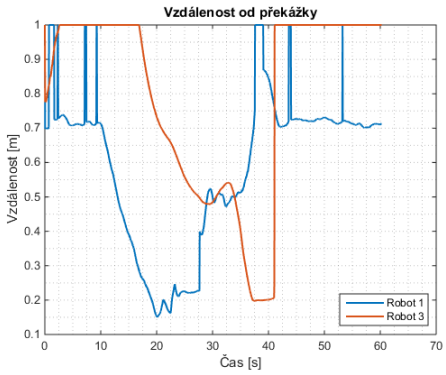
Posledním experimentem je průjezd úzkým prostorem, který testuje schopnost formace změnit dočasně tvar a po projetí úžinou se opět rozmístit na výchozí pozice. Středy šikmých stěn byly umístěny na pozicích [-1.25,2] a [1.25,2] vzhledem k počáteční pozici vedoucího robotu a byly natočeny pod úhlem $-45°$ a $45°$. Za nimi navazovala 1 m dlouhá úžina. Průjezd byl široký 1 m.



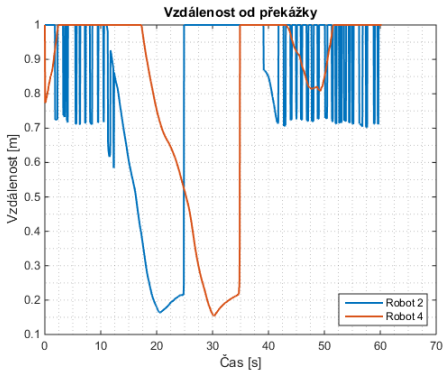
(a) : Trajektorie robotů pro celkový čas jízdy $t = 60.3\,\text{s}$.



(b) : Průběh vzdáleností od výchozích poloh.



(c) : Průběh nejmenších vzdáleností od detekovaných překážek pro roboty 1 a 3.



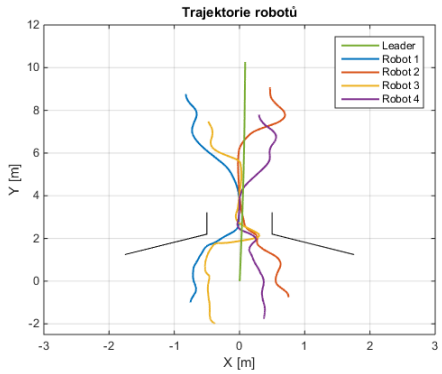
(d) : Průběh nejmenších vzdáleností od detekovaných překážek pro roboty 2 a 4.

Obrázek 3.13: Průběh projetí úžinou pro pružinový algoritmus pro formaci ve tvaru šipky s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.5,-1.25], Robot 2 [0.5,-1], Robot 3 [-1,-2.25], Robot 4 [1,-2].

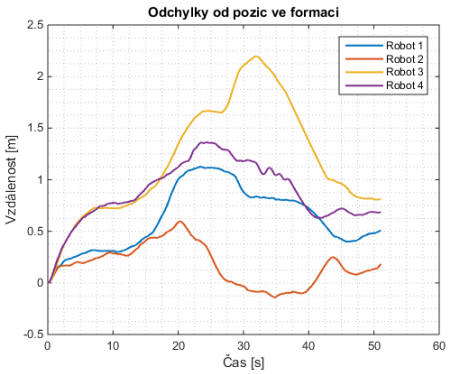
Charakteristické hodnoty - pružiny - šipka				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.68	0.16	0.19	0.30
2	0.41	0.11	0.12	0.14
3	2.46	0.71	1.08	1.29
4	1.26	1.18	0.65	0.69

Tabulka 3.13: Charakteristické hodnoty pro projetí úžinou pro pružinový algoritmus a formaci ve tvaru šipky.

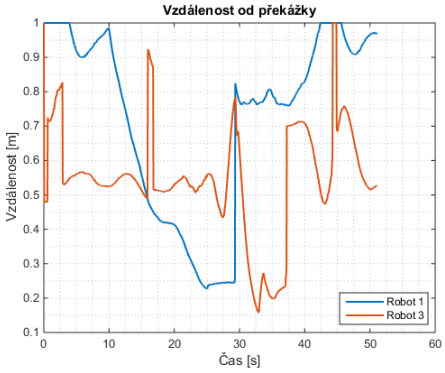
Z obrázku 3.13 je vidět, že robot 3, který projel úžinou jako poslední v pořadí, dosáhl velkého zpoždění za vedoucím robotem a jeho odchylka od výchozí pozice vystoupala až na 2.5 m. K ustálení všech robotů na správné pozice došlo až v čase kolem 60 s.



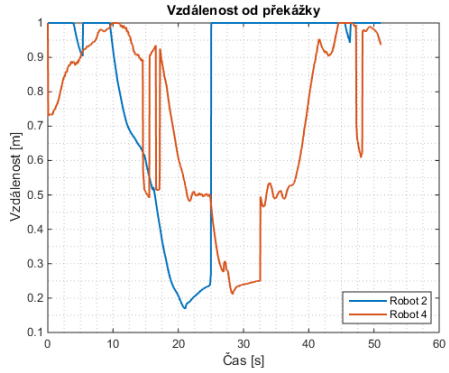
(a) : Trajektorie robotů pro celkový čas jízdy $t = 51.0\,\mathrm{s}$.



(b) : Průběh vzdáleností od výchozích poloh.



(c) : Průběh nejmenších vzdáleností od detekovaných překážek pro roboty 1 a 3.



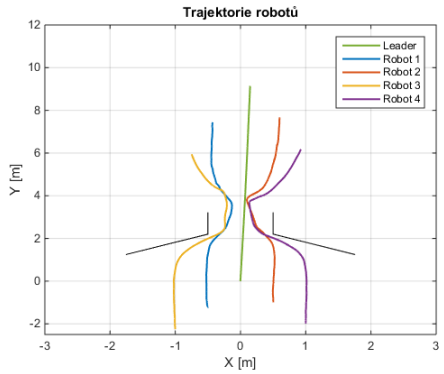
(d) : Průběh nejmenších vzdáleností od detekovaných překážek pro roboty 2 a 4.

Obrázek 3.14: Průběh projetí úžinou pro pružinový algoritmus pro formaci ve tvaru W s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.75,-1], Robot 2 [0.75,-0.75], Robot 3 [-0.375,-2], Robot 4 [0.375,-1.75].

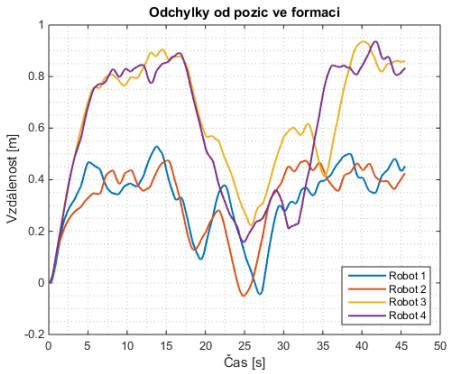
Charakteristické hodnoty - pružiny - W				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	1.13	0.31	0.57	0.62
2	0.60	-0.10	0.17	0.17
3	2.20	0.72	1.02	1.19
4	1.36	0.66	0.80	0.88

Tabulka 3.14: Charakteristické hodnoty pro projetí úžinou pro pružinový algoritmus a formaci ve tvaru W.

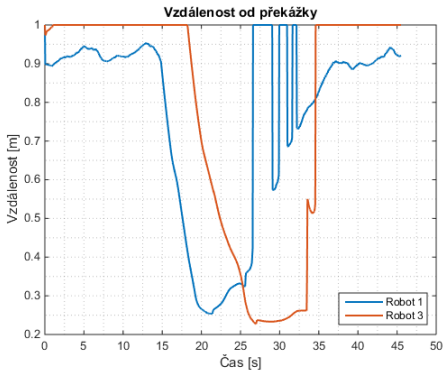
Z obrázku 3.14 je vidět podobný průjezd jako v předchozím případě. Roboty 3 a 4 jsou velmi opožděny za vedoucím robotem. Změnou formace došlo ke zmenšení času ustálení asi o 15 s.



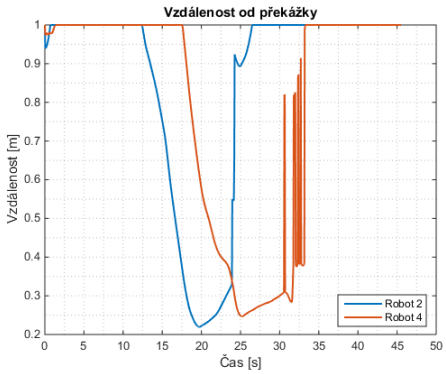
(a) : Trajektorie robotů pro celkový čas jízdy $t = 45.5$ s.



(b) : Průběh vzdáleností od výchozích poloh.



(c) : Průběh nejmenších vzdáleností od detekovaných překážek pro roboty 1 a 3.



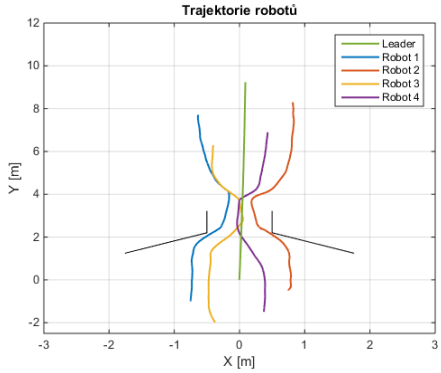
(d) : Průběh nejmenších vzdáleností od detekovaných překážek pro roboty 2 a 4.

Obrázek 3.15: Průběh projetí úžinou pro algoritmus potenciálových polí pro formaci ve tvaru šipky s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.5,-1.25], Robot 2 [0.5,-1], Robot 3 [-1,-2.25], Robot 4 [1,-2].

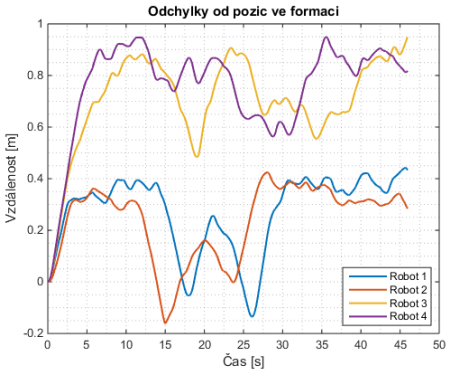
Charakteristické hodnoty - pot. pole - šipka				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.53	0.38	0.36	0.33
2	0.47	0.44	0.37	0.32
3	0.94	0.86	0.67	0.64
4	0.94	0.84	0.75	0.60

Tabulka 3.15: Charakteristické hodnoty pro projetí úžinou pro algoritmus potenciálových polí a formaci ve tvaru šipky.

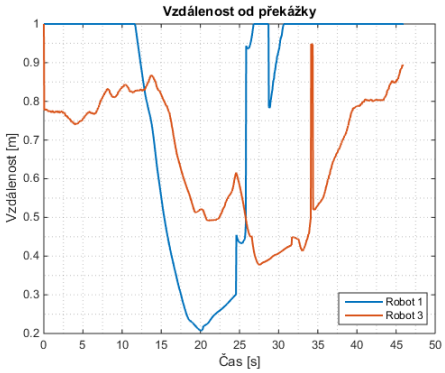
Z obrázku 3.15 je vidět, že průjezd překážkou byl velmi rychlý. K ustálení poloh došlo již asi v čase 35 s. Odchylka od polohy navíc nepřekročila hranici 1 m, což je veliký rozdíl oproti pružinovému algoritmu.



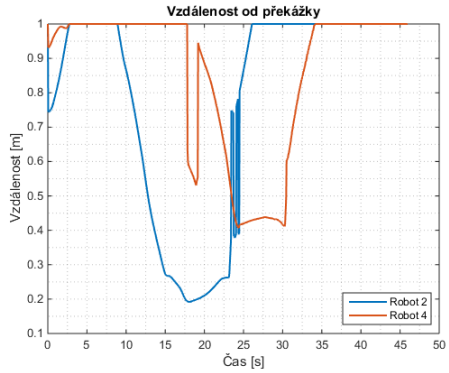
(a) : Trajektorie robotů pro celkový čas jízdy $t = 46.0\,\mathrm{s}$.



(b) : Průběh vzdáleností od výchozích poloh.



(c) : Průběh nejmenších vzdáleností od detekovaných překážek pro roboty 1 a 3.



(d) : Průběh nejmenších vzdáleností od detekovaných překážek pro roboty 2 a 4.

Obrázek 3.16: Průběh projetí úžinou pro algoritmus potenciálových polí pro formaci ve tvaru W s relativními souřadnicemi vzhledem k vedoucímu robotu: Robot 1 [-0.75,-1], Robot 2 [0.75,-0.5], Robot 3 [-0.375,-2], Robot 4 [0.375,-1.5].

Charakteristické hodnoty - pot. pole - W				
Robot	Max [m]	Modus [m]	Medián [m]	Průměr [m]
1	0.44	0.38	0.34	0.27
2	0.42	0.31	0.31	0.14
3	0.95	0.66	0.71	0.71
4	0.95	0.87	0.82	0.76

Tabulka 3.16: Charakteristické hodnoty pro projetí úžinou pro algoritmus potenciálových polí a formaci ve tvaru W.

Z obrázku 3.16 je vidět, že průjezd byl velmi rychlý. Roboty se ustálily opět asi v čase 35 s. Odchylka od polohy opět nepřesáhla hranici 1 m.

3.1.5 Zhodnocení pokusů ze simulátoru

První experiment (kapitola 3.1.1) ukázal základní funkčnost obou rozebraných algoritmů. Mimo jiné průměrnou hodnotou vzdáleností od výchozích poloh formace byla nastavena velikost odchylek, okolo kterých by se měly roboty pohybovat. Pro pružinový algoritmus se tyto odchylky pohybovaly okolo 0.15 m pro první úroveň robotů pro oba typy formace, pro druhou úroveň robotů se odchylka pohybovala okolo 0.65 m pro formaci tvaru šipky a okolo 0.43 m pro formaci tvaru W. Pro algoritmus potenciálových polí se odchylky pro první úroveň robotů pohybovaly okolo 0.41 m pro formaci tvaru šipky a 0.31 m pro formaci tvaru W. Pro druhou úroveň robotů se hodnoty pohybovaly pro obě formace okolo 0.77 m. Pokus mimo jiné ukázal, že algoritmus potenciálových polí má pomalejší reakce na změnu trajektorie vedoucího robotu a tudíž se pohybuje v o něco vyšších hodnotách odchylek.

Druhý pokus (kapitola 3.1.2) ověřil schopnost sledovat vedoucí robot po nelineární trajektorii. Mimo jiné ukázal nedostatek v obou algoritmech, díky kterému dochází ke zkracování obloukových trajektorií. Předchozí je způsobeno okamžitou reakcí na změnu trajektorie vedoucího robotu a tím, že algoritmus nebere ohled na zpoždění poloh vedoucího robotu a sledujícího robotu. Kromě tohoto nedostatku byla úspěšně ověřena schopnost sledovat vedoucí robot po netriviální dráze, která je vidět na obrázku 3.5 a a je vyznačena zelenou barvou.

Třetí pokus (kapitola 3.1.3), kterým bylo objetí kruhového sloupu, byl ve všech čtyřech případech úspěšně dokončen. Pružinový algoritmus měl oproti potenciálovým polím větší problémy při ustálení na výchozích pozicích. Maximální hodnota odchylky se pohybovala v hodnotách téměř o 0.5 m více než u potenciálových polí. Minimální vzdálenost od překážky u potenciálových polí se pohybovala nejníže okolo 0.26 m, což lze považovat stále za bezpečné.

Pro pružinový algoritmus se minimální vzdálenost pohybovala okolo 0.35 m. V případě formace W pro pružinový algoritmus minimální hodnota klesla až na 0.16 m. Tento pokles byl způsoben přiblížením robotu 4 pod úhlem, který byl mimo záběr laserového dálkoměru robotu 1. Při otočení robotu 1 kvůli vyhnutí se překážce se robot 4 objevil přímo před robotem 1, ani v této situaci však nedošlo ke kolizi robotů. Lze říci, že v tomto pokusu si vedl lépe algoritmus potenciálových polí, který měl v obou případech použitých algoritmů odchylky od výchozích poloh a rychleji se dokázal ustálit zpět na pozicích určených z prvního pokusu.

Charakteristické hodnoty třetího pokusu			
Algoritmus/ formace	Max. odchylka [m]	Čas ustálení [s]	Min. vzdálenost od překážky [m]
Pružiny/šipka	1.95	45	0.33
Pružiny/W	1.85	44	0.16 (0.38)
Pot. pole/šipka	1.37	38	0.23
Pot. pole/W	1.41	31	0.26

Tabulka 3.17: Sledovaná kritéria pro objíždění sloupu.

Pro poslední pokus (kapitola 3.1.4) platí podobné výsledky jako v předchozím případě. Maximální odchylky od poloh se pohybovaly okolo 2.4 m pro pružinový algoritmus, což je výrazně víc než u potenciálových polí, kde odchylka nepřesáhla 1 m. Rychlost ustálení byla opět menší v případě potenciálových polí a to nejméně o 7 s. Minimální vzdálenost od překážek byla pro všechny čtyři případy velmi podobná a pohybovala se kolem hodnoty 0.20 m.

Charakteristické hodnoty čtvrtého pokusu			
Algoritmus/ formace	Max. odchylka [m]	Čas ustálení [s]	Min. vzdálenost od překážky [m]
Pružiny/šipka	2.46	60	0.15
Pružiny/W	2.20	47	0.16
Pot. pole/šipka	0.94	40	0.22
Pot. pole/W	0.95	40	0.19

Tabulka 3.18: Sledovaná kritéria pro průjezd úžinou.

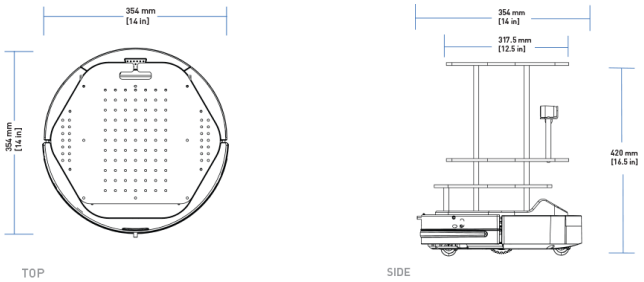
Algoritmus potenciálových polí, i přes menší citlivost na změny v trajektorii vedoucího robotu a pomalejší reakce, dosáhl obecně lepších výsledků než pružinový algoritmus. Nicméně oba algoritmy se ukázaly schopné formaci udržet.

Kapitola 4

Realizace

4.1 Reálný robot

Pro reálné testování byl použit robot Turtlebot [8]. Pro jeho řízení byl zvolen systém ROS (Robot Operating System), neboť robot je vyvíjen přímo ve spolupráci s vývojáři tohoto systému. Tento robot je schopen vyvinout maximální rychlost až 0.65 m/s a rychlost otáčení až 180 °/s. Sám váží 6.3 kg a je schopen přepravit dalších 5 kg užitečné zátěže.



(a) : Rozměry těla robota Turtlebot shora.

(b) : Výška a šířka podpůrné konstrukce pro senzory.

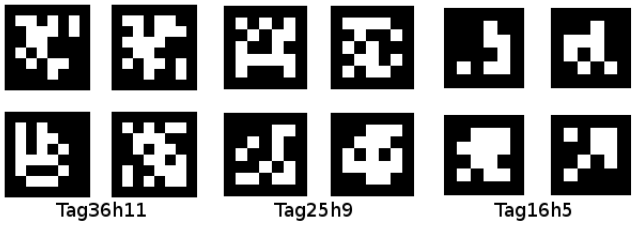
Obrázek 4.1: Rozměry robotu Turtlebot. Převzato z [21] 15.5.2017.

Na robot byl připevněn laserový dálkoměr Hokuyo URG-04LX-UG01 s dosahem až 5.6 m. Dálkoměr měří v horizontální rovině v šířce 180° s nastavitelnou velikostí kroku, která je dostatečně malá, aby nám zajistila kvalitní data pro collision avoidance. Data jsou předávána přes USB port pomocí systému ROS. Napájen je napětím 5 V z USB.

Obrázek 4.2: Laserový dálkoměr Hokuyo URG-04LX-UG01. Převzato z [22] 15.5.2017.

Dále byl robot vybaven kamerou Basler daA1280-54um s rychlostí snímkování až 54 fps a rozlišením 1280 px x 960 px.

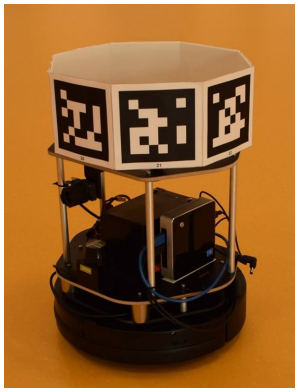
Pro sledování robotů, včetně vedoucího, byly zvoleny značky Apriltags [14]. Díky jejich rozlišitelnosti je možné rozlišit jednotlivé roboty. Tyto značky mimo jiné umožňují určit všech 6 stupňů volnosti. Takto je pak možné určit úhel, pod kterým je daný robot sledován. Značky byly zvoleny z řady 36h11 s rozměrem 11 cm x 11 cm [14]. Každý z robotů je vybaven 8-úhelníkovým prstencem pokrytým těmito značkami.



Obrázek 4.3: Ukázka značek Apriltags. Převzato z [15] 15.5.2017.

4.2 Popis reálných experimentů

Pružinový algoritmus vytvořený v rámci této práce byl testován na reálných robotech (v rámci práce [8]) sestavených z výše uvedených komponent. Z důvodu nedostatku robotů byl vedoucí robot nahrazen člověkem, který osmiúhelníkovým prstencem se značkami simuloval trajektorii vedoucího robotu. Na každém robotu byla připevněna pouze jedna kamera, tudíž byly možnosti formace značně omezené. Pro reálné sledování značek bylo nutné kameru natočit přibližně o 10°, aby se pozice značek při ustáleném přímočarém pohybu nacházela přibližně uprostřed.



Obrázek 4.4: Sestavený robot se všemi částmi. Převzato z [8] 17.5.2017.

Algoritmus byl testován pro sedm různých situací [8]:

- Jízda po přímce.
- Jízda po nelineární trajektorii.
- Objezd rohové překážky.
- Objezd kruhového sloupu.
- Průjezd zúženým místem.
- Jízda proti překážce ve tvaru klínu.
- Jízda proti překážce tvaru písmene L.

Algoritmus obstál v 5 ze 7 testů [8]. V žádném testu nedošlo ke kolizi mezi roboty či s překážkou. Během jízdy nedošlo k přiblížení na menší vzdálenost než 0.20 m. Robot nesplnil podmínky testu pro objetí kruhového sloupu. Ten sice objel, ovšem s velkou odchylkou [8]. Robot dále nedokázal objet překážku tvaru L. Tento případ byl jediný, kdy došlo k roztržení formace.

Algoritmus byl vzhledem k omezeným možnostem sledování vyhodnocen jako dostatečně kvalitní [8]. Pro roztržení formace byla potřeba složitá překážka.

(a) : Reálný pokus - jízda za roh.

(b) : Reálný pokus - jízda okolo sloupu.

(c) : Reálný pokus - průjezd úžinou.

(d) : Reálný pokus - jízda proti klínu.

(e) : Reálný pokus - jízda proti překážce tvaru L.

Obrázek 4.5: Foto z reálného testování. Převzato z [8] 17.5.2017.

Závěr

V první kapitole bylo představeno několik principů, kterých se využívá po celém světě při vytváření multiagentních systémů. Tyto systémy se stávají nenahraditelnou složkou strojní automatizace a to hlavně z důvodu rychlosti a komplexnosti řešení, které nabízejí. Všechny zde zmíněné algoritmy využívají známých relativních poloh svých sousedů. Podrobně zde byly zpracovány dva z těchto algoritmů. Prvním byl pružinový algoritmus využívající virtuálních pružin, které jsou obdobou klasické mechanické pružiny. Druhým byl algoritmus využívající potenciálových polí. Pro oba algoritmy byl podrobně rozebrán postup udržování formace během následování vedoucího robotu.

Ve druhé kapitole byl podrobně rozpracován princip vyhýbání se překážkám pro oba algoritmy představené v první kapitole. Obě metody následování vedoucího robotu byly i s funkcionalitou vyhýbání se překážkám implementovány v jazyce Lua v simulačním prostředí V-REP. V této kapitole byl rozebrán princip Kalmanova filtru, který byl využit při reálném testování jednoho z navržených algoritmů. Jeho úkolem bylo mimo jiné predikovat polohu vedoucího robotu v případě jeho ztráty z obrazu kamery. Sledující robot tak byl schopen překonat překážky, při kterých se vedoucí robot ztratil ze záběru kamery. Následně se díky predikci polohy vedoucího robotu dokázal opět zařadit do formace.

Ve třetí kapitole byly oba implementované algoritmy testované dle různých kritérií. Prvním testem byla jízda po přímce, která testovala základní schopnost udržování formace. Stanovila odchylky od polohy, které byly použity jako referenční hodnoty při vyhodnocení formace jako ustálené po projetí oblasti s překážkami. Druhým pokusem byla otestována schopnost sledovat vedoucí robot po nelineární trajektorii. Ten ověřil pokročilou schopnost následování vedoucího robotu zbytkem formace. V dalších experimentech byla testována

funkcionalita vyhýbání se překážkám. První překážkou byl sloup postavený v dráze robotu. Druhou překážkou byla úžina, která ověřovala schopnost dočasné změny tvaru formace a její následné narovnání do výchozího tvaru. Oba testy s překážkami ukázaly, že pružinový algoritmus dosahoval vyšších hodnot odchylek od výchozích poloh a také delších časových intervalů pro ustálení na pozicích stanovených prvním pokusem oproti algoritmu potenciálových polí. V testu, při průjezdu úžinou, měly roboty s použitým algoritmem potenciálových polí velmi malou odchylku od výchozích poloh, která dokonce nepřesáhla hranici jednoho metru. V žádném z pokusů také nedošlo ke srážce robotů s překážkou nebo mezi sebou. Oba algoritmy se ukázaly být schopné řídit formaci robotů, včetně integrované funkcionality vyhýbání se překážkám na základě známých relativních poloh, avšak algoritmus potenciálových polí se ukázal lepší, protože dosahoval menších odchylek od výchozích poloh a kratších časů pro ustálení.

Oba implementované algoritmy splnily cíle této práce. Avšak do budoucna by bylo vhodné, aby řídící algoritmus bral v úvahu časové zpoždění poloh mezi sledujícím robotem a vedoucím robotem, aby při nelineární trajektorii vedoucího robotu nedocházelo ke zkracování obloukových trajektorií a tím ke zmenšení plochy, kterou je formace schopna pokrýt. Dále by bylo vhodné vytvořit komunikační síť mezi členy formace, aby tak byly schopny sdílet informace o svých polohách a lépe zvolit vhodnou trajektorii.

Literatura

[1] V-rep, coppelia robotics. `http://www.coppeliarobotics.com/`.

[2] Stavový model a Kalmanův filtr. `https://www.fd.cvut.cz/personal/provipav/Stochastika/Materialy-test4/Stav.pdf`, 2013.

[3] Kalman filter. `https://github.com/hmartiro/kalman-cpp`, 2014.

[4] Jan Carlo Barca, A Sekercioglu, and A Ford. Controlling formations of robots with graph theory. *Intelligent Autonomous Systems 12*, pages 563–574, 2013.

[5] Laura Barnes, Wendy Alvis, MaryAnne Fields, Kimon Valavanis, and Wilfrido Moreno. Swarm formation control with potential fields formed by bivariate normal functions. In *Control and Automation, 2006. MED'06. 14th Mediterranean Conference on*, pages 1–7. IEEE, 2006.

[6] Laura Barnes, MaryAnne Fields, and Kimon Valavanis. Unmanned ground vehicle swarm formation control using potential fields. In *Control & Automation, 2007. MED'07. Mediterranean Conference on*, pages 1–8. IEEE, 2007.

[7] Carlos Bentes and Osamu Saotome. Dynamic swarm formation with potential fields and a* path planning in 3d environment. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, pages 74–78. IEEE, 2012.

[8] Matěj Beránek. Experimentální prostředí pro skupinovou robotiku. Bachelor thesis, Czech Technical University in Prague, 2017.

[9] Radek Burda. Simulace davu metodou boids. *FIT VUT v Brně*, pages 5–7, 2013.

[10] Samitha W Ekanayake and Pubudu N Pathirana. Formations of robotic swarm: An artificial force based approach. *International journal of advanced robotic systems*, 6(1):7, 2009.

[11] Saing Paul Hou, Chien-Chern Cheah, and Jean-Jacques E Slotine. Dynamic region following formation control for a swarm of robots. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1929–1934. IEEE, 2009.

[12] Donghwa Jeong and Kiju Lee. Dispersion and line formation in artificial swarm intelligence. *arXiv preprint arXiv:1407.0014*, 2014.

[13] Taufik Khuswendi, Hilwadi Hindersah, and Widyawardana Adiprawita. Uav path planning using potential field and modified receding horizon a* 3d algorithm. In *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*, pages 1–6. IEEE, 2011.

[14] Martin Žákovec. Vizuální navigace formace mobilních robotů. Bachelor thesis, Czech Technical University in Prague, 2017.

[15] April Robotics Laboratory. Apriltag. `https://april.eecs.umich.edu/software/apriltag.html`.

[16] Chien-Chou Lin, Kun-Cheng Chen, Po-Yuan Hsiao, and Wei-Ju Chuang. Motion planning of swarm robots using potential-based genetic algorithm. *International Journal of Innovative Computing, Information and Control ICIC*, 9:305–318, 2013.

[17] Iñaki Navarro and Fernando Matía. An introduction to swarm robotics. *ISRN Robotics*, 2013, 2012.

[18] Jacques Penders. Robot swarming applications. 2007.

[19] Craig W Reynolds. Boids. `http://www.red3d.com/cwr/boids/`.

[20] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics*, 21(4):25–34, 1987.

[21] Clearpath Robotics. Turtlebot 2. `https://www.clearpathrobotics.com/turtlebot-2-open-source-robot/`.

[22] RobotShop. Hokuyo urg-04lx-ug01. `http://www.robotshop.com/en/hokuyo-urg-04lx-ug01-scanning-laser-rangefinder.html`.

[23] Angie Shia. Survey of swarm robotics techniques–a tutorial. *Special Topics Chair IEEE RAS Region*, 6.

[24] Brian Shucker and John K Bennett. Virtual spring mesh algorithms for control of distributed robotic macrosensors. *University of Colorado at Boulder, Technical Report CU-CS-996-05*, 2005.

[25] Antonín Vojáček. Co je to Kalmanova filtrace. http://automatizace.hw.cz/clanek/2007042901, 2007.

[26] Zbyněk Winkler. Měření rychlosti - kalmanův filtr. http://robotika.cz/guide/filtering/en, 2005.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Martin N o v o t n ý

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Robustní postupy řízení formací robotů

Pokyny pro vypracování:

1. Seznamte se s algoritmy a postupy pro řízení formací pozemních mobilních robotů na základě měření relativní vzdálenosti a směru mezi jednotlivými roboty.
2. Vyberte nejméně 2 postupy vhodné pro řízení formace založené na měření relativní vzdálenosti a orientace mezi jednotlivými roboty a tyto postupy implementujte. Do řešení zahrňte výskyt neprostupných překážek v pracovním prostředí a zahrňte funkcionalitu předcházení kolizím.
3. Implementaci vybraných postupů proveďte v simulačním prostředí VREP, ověřte jejich chování, navrhněte vhodná kritéria hodnocení a algoritmy kvantitativně srovnejte z hlediska kvality udržování formace (stability, přesnosti a schopnosti obnovení po provozním selhání). Průběh simulací dokumentujte krátkým videozáznamem.

Seznam odborné literatury:

[1] Donghwa Jeong, and Kiju Lee, "Dispersion and Line Formation in Artificial Swarm Intelligence," Case Western Reserve University, 2014
[2] B. Shucker, J. K. Bennett, "Virtual spring mesh algorithms for control of distributed robotic macrosensors," Department of computer science university of Colorado in Boulder, May 2005
[3] L. Barnes, W. Alvis, M. Fields, K. Valavanis, and W. Moreno, "Swarm Formation Control with Potential Fields Formed by Bivariate Normal Function," Control and Automation, 2006

Vedoucí bakalářské práce: Ing. Libor Přeučil, CSc.

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2017